ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Algoritmické obchodování na burze

Bc. Jiří Fischer

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jiří Fischer

Informatika

Název práce

Algoritmické obchodování na burze

Název anglicky

Algorithmic trading on the stock exchange

Cíle práce

Cílem práce je návrh, implementace a testování algoritmu pro intra denní obchodování momentových akcií. Hlavním úkolem algoritmu bude spočívat ve vyhledání nejobchodovanějších akcií na burze v daný den u kterých následně detekuje nákupní a prodejní signály pomocí nově se tvořících jednoduchých svíčkových formací a námi zvolených pravidel, indikátorů a oscilátorů. Pravidla, indikátory a oscilátory budou vybrány na základě podrobného zpětného testování pomocí historických dat a následně podložené statistickou analýzou. V práci bude také navržena a implementována desktopová aplikace s GUI, která bude propojena s obchodovací platformou Interactive Brokers. Aplikace bude sloužit ke správě, kontrole, přizpůsobení a nastavení algoritmu v reálném čase.

Metodika

První, teoretická část této diplomové práce bude sloužit k uvedení do problematiky a popisu důležitých termínů týkajících se obchodování na burze, algoritmického obchodování, vyhodnocení výkonosti algoritmu a použitých metod, indikátorů, oscilátorů a použitých programovacích nástrojích. V druhé části bude navržen algoritmus, který bude detekovat jednoduché svíčkové formace pomocí zjištění vyšších výšek, či nižšího minima u nejobchodovanějších akcií v daný den. Ve třetí části bude identifikace nejefektivnějších pravidel, indikátorů a oscilátory pro nákup a prodej na základě zpětného testování pomocí historických dat skrze implementovaný algoritmus detekování formací z druhé části. GUI aplikace, která bude propojena s obchodovací platformou Interactive Brokers pro správu, kontrolu, přizpůsobení a nastavení algoritmu v reálném čase. Závěry budou implementovány do finálního algoritmu, který bude otestován v reálném prostředí pomocí navrhnuté desktop aplikace, a výkonnost algoritmu bude hodnocena na základě Sharpeho poměru.

Doporučený rozsah práce

60-80 stran

Klíčová slova

burza, systém, algoritmické obchodování, automatické obchodování, svíčkové formace, hybnost, likvidita, autonomní

Doporučené zdroje informací

Cartea, Álvaro, Jaimungal, Sebastian a Penalva, José. Algorithmic and high-frequency trading. 1. Cambridge : Cambridge University Press, 2015. str. 360. 978-1-107-09114-6.

Ernest, Chan. Machine Trading: Deploying Computer Algorithms to Conquer the Markets. 1. Hoboken : John Wiley & Sons, Inc., 2017. str. 264. 978-1-119-21960-6.

Ernest, Chan. Quantitative Trading: How to Build Your Own Algorithmic Trading Business. 2. Hoboken : John Wiley & Sons, Inc., 2021. str. 208. 978-1-119-80006-4.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 01. 04. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Algoritmické obchodování na burze" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 03. 2023 _____

Poděkování

Nejprve bych rád upřímně poděkoval doc. Ing. Vojtěchu Merunkovi, Ph.D. za jeho podporu při výběru tohoto zajímavého tématu. Jeho trpělivost a odborné vedení byly klíčovými faktory pro úspěšné dokončení mé diplomové práce.

Dále bych rád poděkoval skupině JP Morgan Chase za jejich online stáž, která mi poskytla možnost pracovat na velké šířce problematiky na kapitálových trzích. Taktéž bych rád poděkoval společnosti AlphaVantage za poskytnutí výzkumného API klíče, což mi umožnilo získat nezbytná data pro tuto práci.

Nakonec bych chtěl poděkovat své rodině a přátelům za podporu a povzbuzení během mého studia a psaní této práce.

Algoritmické obchodování na burze

Abstrakt

V poslední dekádě je tempo automatizace v odvětví intradenního obchodování akcií za účelem maximalizace zisku při co nejnižší rizikovosti posedlé tématy jako je algoritmické obchodování a strojové učení. Odborníci z oboru odhadují, že až 75 % denního obchodování je připsáno algoritmickému obchodování. Pasivní, či automatizované obchodování nabízí spoustu výhod zejména v efektivnosti obchodování skrze řízení ceny, velikosti a časování obchodů v reálném čase v reakci na výkyvy v akciovém trhu. Dále nabízí spoustu dalších benefitů jako je transparentnost, konzistentnost, kompetitivní výhoda, bezpečnost, přístup a škálovatelnost. Predikce akciového trhu je v dnešní době nesmírně zajímavá záležitost jak pro finanční instituce, tak jednotlivce zejména kvůli vysokým ziskovým maržím. Většina úspěšných nákupů nebo prodejů je zpravidla spojována se změnou trendu. Precizní odhad kolísání cen společností hraje důležitou roli v nákupních a prodejních aktivitách.

Klíčová slova: Burza, Systém, Algoritmické obchodování, Automatické obchodování, svíčkové formace, hybnost, likvidita

Algorithmic trading on the stock exchange

Abstract

In the last decade, the pace of automation in the intraday stock trading industry to maximize profits while minimizing risk has led to topics such as algorithmic trading and machine learning. Experts estimate that up to 75% of daily trading is attributed to algorithmic trading. Passive or automated trading offers many advantages, especially in trading efficiency through real-time management of price, size, and timing of trades in response to fluctuations in the stock market. It also offers many other benefits such as transparency, consistency, competitive advantage, security, accessibility, and scalability. Stock market prediction is an incredibly interesting matter for financial institutions and individuals, especially due to high profit margins. Most successful purchases or sales are usually associated with a change in trend. Precise estimation of company stock price fluctuations plays an important role in buying and selling activities.

Keywords: Stock exchange, System, Algorithmic trading, Automated trading, Candlestick patterns, Momentum, Liquidity

Obsah

1 Úvod 10
2 cíl práce a metodika 11
2.1 Cíle práce: 11
2.2 Metodika 11

3 TEORETICKÁ ČÁST 12
3.1 Shrnutí mnoha technik algoritmického obchodování 12
3.1.1 přehled základních charakteristik algoritmického obchodování 12
3.1.2 Vytváření nástrojů a technik pro algoritmické obchodování 13
3.1.3 Druhy algoritmického obchodování 14
3.2 Výhody a nevýhody automatizovaného obchodování 16
3.2.1 Výhody 16
3.2.2 Nevýhody 17
3.2.3 Proměnné ovlivňující algoritmické obchodní metody 20
3.3 Algoritmické obchodní metody a technické nástroje analýzy 22
3.3.1 Formace svíčkových grafů 22
3.3.2 klouzavé průměry 22
3.3.3 Trendové čáry a kanály 23
3.3.4 Úrovně podpory a odporu 23
3.3.5 Grafické tvary 24
3.3.6 Technické indikátory 24
3.4 Metriky pro hodnocení algoritmických obchodních strategií 26
3.4.1 Objektivní ukazatele výkonu 27
3.4.2 Metriky výkonnosti, které zohledňují riziko 27
3.4.3 Sledování výkonnosti v průběhu času 28
3.4.4 Srovnání s benchmarky 30
3.4.5 Analytické postupy a ověření modelu 31

4 Praktická část 33
4.1 Návrh a přístup k výzkumu 33
4.1.1 Účel výzkumu 33
4.1.2 Výzkumné otázky 33
4.1.3 Výzkumné hypotézy 34
4.1.4 Výzkumná metodologie 34
4.1.5 Omezení výzkumu 35
4.2 Metody sběru a analýzy dat 35
4.2.1 Zdroje dat 35
4.2.2 Postupy sběru dat 36
4.2.3 Techniky analýzy dat 36
4.2.4 Validace dat 37

4.3 Vývoj a implementace algoritmu 37
4.3.1 Vývoj algoritmu 38
4.3.2 Implementace algoritmu 38
4.3.3 Programovací jazyk a nástroje 39
4.4 Popis vybraných technických analýzových nástrojů 39
4.4.1 Klouzavé průměry 39
4.4.2 Relativní síla index (RSI) 40
4.4.3 Stochastický oscilátor 40
4.4.4 Bollingerhova pásma 40
4.4.5 MACD indikátor 41
4.5 Backtestování a optimalizace algoritmu pomocí historických dat 41
4.5.1 Postup backtestování 41
4.5.2 Optimalizační techniky 42
4.6 Vývoj aplikace s grafickým rozhraním a integrace 43
4.6.1 Návrh uživatelského rozhraní 43
4.6.2 Funkce aplikace 43
4.6.3 Integrace s platformou Interactive Brokers 44
4.6.4 Reálné datové toky 44
4.6.5 Systém monitorování a výstrah v reálném čase 45
4.6.6 Postup úpravy algoritmu 45

5 softwarové řešení 46
5.1 Data 46
5.2 Statistická analýza 49
5.3 GUI 50
5.4 Backtesting – broker propojení 55

6 Výsledky a backtestování 57
6.1 1. backtestování 57
6.2 2. backtestování 58
6.3 3. backtestování 60
6.4 4. backtestování 61
6.5 5. backtestování 63
6.6 Výsledky 64
6.7 Obchodování v realném čase 65

7 Závěr 68
8 Literatura 69
9 Seznam obrázků a tabulek 74
9.1 Seznam obrázků 74
9.2 Seznam tabulek 75

10 Přílohy 76

1 ÚVOD

V posledních letech se finanční svět stále více přesouvá směrem k automatizaci a využití moderních technologií, které přinášejí nové možnosti a efektivitu v oblasti obchodování. Významným trendem, který se v poslední dekádě objevil v odvětví intradenního obchodování, je algoritmické obchodování a strojové učení. Díky nim je možné maximalizovat zisk při minimalizaci rizika a zároveň udržet krok s rychle se měnícím trhem. Odhaduje se, že až 75 % denního obchodování je nyní řízeno algoritmami, což ukazuje na význam a potenciál této metody.

Tato diplomová práce se zaměřuje na návrh, implementaci a testování algoritmu pro intradenní obchodování momentových akcií. Hlavním úkolem algoritmu bude vyhledání nejobchodovanějších akcií na burze v daný den a detekce nákupních a prodejních signálů pomocí jednoduchých svíčkových formací a zvolených pravidel, indikátorů a oscilátorů. Tyto prvky budou vybrány na základě podrobného zpětného testování s využitím historických dat a následně podloženy statistickou analýzou.

Součástí práce bude také návrh a implementace desktopové aplikace s grafickým uživatelským rozhraním (GUI), která bude propojena s obchodovací platformou Interactive Brokers. Aplikace umožní správu, kontrolu, přizpůsobení a nastavení algoritmu v reálném čase.

V teoretické části práce budou představeny základní pojmy týkající se obchodování na burze, algoritmického obchodování, vyhodnocení výkonosti algoritmu a použitých metod, indikátorů, oscilátorů a programovacích nástrojů. Následně bude prezentován návrh algoritmu, testování a identifikace nejefektivnějších pravidel, indikátorů a oscilátorů, a implementace GUI aplikace. Závěr práce bude věnován testování finálního algoritmu v reálném prostředí pomocí navrhnuté desktopové aplikace a hodnocení výkonnosti algoritmu na základě Sharpeho poměru.

2 CÍL PRÁCE A METODIKA

2.1 CÍLE PRÁCE:

Cílem práce je návrh, implementace a testování algoritmu pro intra denní obchodování momentových akcií. Hlavním úkolem algoritmu bude spočívat ve vyhledání nejobchodovanějších akcií na burze v daný den u kterých následně detekuje nákupní a prodejní signály pomocí nově se tvořících jednoduchých svíčkových formací a námi zvolených pravidel, indikátorů a oscilátorů. Pravidla, indikátory a oscilátory budou vybrány na základě podrobného zpětného testování pomocí historických dat a následně podložené statistickou analýzou. V práci bude také navržena a implementována desktopová aplikace s GUI, která bude propojena s obchodovací platformou Interactive Brokers. Aplikace bude sloužit ke správě, kontrole, přizpůsobení a nastavení algoritmu v reálném čase.

2.2 METODIKA

První, teoretická část této diplomové práce bude sloužit k uvedení do problematiky a popisu důležitých termínů týkajících se obchodování na burze, algoritmického obchodování, vyhodnocení výkonosti algoritmu a použitých metod, indikátorů, oscilátorů a použitých programovacích nástrojích. V druhé části bude navržen algoritmus, který bude detekovat jednoduché svíčkové formace pomocí zjištění vyšších výšek, či nižšího minima u nejobchodovanějších akcií v daný den. Ve třetí části bude identifikace nejefektivnějších pravidel, indikátorů a oscilátory pro nákup a prodej na základě zpětného testování pomocí historických dat skrze implementovaný algoritmus detekování formací z druhé části. GUI aplikace, která bude propojena s obchodovací platformou Interactive Brokers pro správu, kontrolu, přizpůsobení a nastavení algoritmu v reálném čase. Závěry budou implementovány do finálního algoritmu, který bude otestován v reálném prostředí pomocí navrhnuté desktop aplikace, a výkonnost algoritmu bude hodnocena na základě Sharpeho poměru.

3 TEORETICKÁ ČÁST

3.1 SHRNUTÍ MNOHA TECHNIK ALGORITMICKÉHO OBCHODOVÁNÍ

Burza cenných papírů prošla v posledních několika desetiletích značnou transformací, především díky algoritmickému obchodování, rychle se rozvíjejícímu sektoru finančního průmyslu. Tato kapitola důkladně zkoumá různé aspekty algoritmického obchodování, aby poskytla čtenářům jasnou představu o základních myšlenkách, základní technologii a různých druzích algoritmických obchodních strategií používaných účastníky trhu.

3.1.1 PŘEHLED ZÁKLADNÍCH CHARAKTERISTIK ALGORITMICKÉHO OBCHODOVÁNÍ

Algoritmické obchodování, často nazývané automatizované obchodování nebo „algo-trading“, se odkazuje na použití počítačových algoritmů a statistických modelů k provádění finančních transakcí na trhu s minimální nebo žádnou účastí člověka. Tyto algoritmy využívají cenových rozdílů a tržních neefektivit k rychlé analýze tržních dat, identifikaci obchodních příležitostí a provádění objednávek.

Některé z obecných charakteristik algoritmického obchodování zahrnují následující:

Rychlost: Jelikož algoritmické obchodní systémy mohou zpracovávat obrovské množství dat a provádět obchody v milisekundách, výrazně předčí lidské obchodování. Díky rychlému provádění mohou obchodníci profitovat z arbitrážních příležitostí a efemérních tržních příležitostí.

Přesnost: Automatizované obchodní systémy snižují možnost lidské chyby tím, že provádějí transakce s vysokou mírou přesnosti. Můžou také spravovat současně několik obchodních strategií a instrumentů, což umožňuje větší diverzifikaci portfolia.

Škálovatelnost: Algoritmické obchodní systémy jsou zvláště škálovatelné, protože mohou zvládat velké množství objednávek, strategií a pozic. Tato škálovatelnost je klíčová pro obchod vysoké frekvence a masivní institucionální investory, kteří potřebují efektivní nástroje pro správu svých obrovských objemů obchodování.

Požadavky a tolerance k riziku konkrétních obchodníků nebo institucí mohou být zohledněny v algoritmech algoritmického obchodování. Tato adaptabilita umožňuje vytváření unikátních obchodních strategií, které zohledňují širokou škálu finančních cílů a tržních podmínek.

Nákladová efektivnost: Snížením tzv. "slippage" a potřeby manuální intervence může algoritmické obchodování vést k nižším transakčním nákladům a vyšší kvalitě provedení obchodu. Tato nákladová efektivita by byla zvláště výhodná pro obchodníky s vysokou frekvencí a velké institucionální investory, kteří chtějí snížit náklady na obchodování.

Obchodování bez emocí: Použití algoritmů při obchodování snižuje vliv lidských emocí na obchodní rozhodnutí, což vede k logičtějším a nezaujatějším rozhodnutím. To může být zvláště užitečné v extrémně volatilních nebo stresových tržních podmínkách, kde mohou být lidští obchodníci náchylnější k impulsivním nebo iracionálním reakcím.

V následujících částech se tato kapitola zabývá různými typy algoritmických obchodních technik používaných účastníky trhu, stejně jako technologickými základy pro algoritmické obchodování. Tato podrobná analýza je nezbytná k pochopení nuancí a složitosti této neuvěřitelně sofistikované formy obchodování, která neustále mění mezinárodní finanční trhy.

3.1.2 VYTVÁŘENÍ NÁSTROJŮ A TECHNIK PRO ALGORITMICKÉ OBCHODOVÁNÍ

Operace obchodování mohou být automatizovány díky pokrokům v počítačových a telekomunikačních technologiích v 80. letech. To byl vznik algoritmických obchodních technologií a postupů (Leinweber, 2009). Následující jsou významné zlomky v historii vývoje algoritmického obchodování:

Elektronické komunikační sítě (ECN) byly vytvořeny koncem 80. let a začátkem 90. let, aby umožnily účastníkům trhu obchodovat navzájem bez zprostředkovatelů (Algorithmic trading and the market for liquidity, 2013). Tento vynález přispěl k rozvoji algoritmického obchodování tím, že nabídl základy nezbytné pro automatické umísťování a provádění objednávek.

Vznik kvantitativního obchodování v 90. letech otevřel cestu k vývoji sofistikovaných algoritmických obchodních strategií (Chan, 2008). Kvantitativní obchodníci využívali matematické modely a nejmodernější výpočetní techniky k identifikaci a využití tržních neefektivit, což otevřelo dveře pro vývoj složitějších algoritmických strategií.

Vysokofrekvenční obchodování (HFT): V prvních letech 21. století se objevilo HFT, které využívalo ultra-nízko-latentní technologie k provádění obchodů v mikrosekundách (Aldridge,

2010). Firmy specializující se na HFT využívaly svou výhodu v rychlosti k využití příležitostí pro arbitráž a tržní neefektivitu, než by mohli konkurenti zareagovat.

Zvýšená dostupnost velkých dat a pokrok v oblasti strojového učení a umělé inteligence v 10. letech vedly k vývoji obchodních algoritmů s neustále se zvyšující sofistikací (Machine learning in finance: The case of deep learning for option pricing, 2020). Tyto systémy pomáhaly obchodníkům vytvářet informovanější rozhodnutí tím, že analyzovaly tržní data a používaly prediktivní analytiku a rozpoznávání vzorů.

Bankovní sektor využívá distribuovanou technologii ledgers (DLT) a cloudové výpočty v 20. letech (Tapscott, 2016). Těmito technologiemi se umožňuje vývoj decentralizovaných obchodních platforem, které zlepšují škálovatelnost a bezpečnost algoritmických obchodních systémů.

3.1.3 DRUHY ALGORITMICKÉHO OBCHODOVÁNÍ

(The science of algorithmic trading and portfolio management, 2013) kategorizuje algoritmické obchodní techniky do následujících skupin:

Strategie tvorby trhu (market-making): Market makerové využívají algoritmického obchodování k nabídnutí cenových kotací pro finanční produkty s cílem zvýšit tržní likviditu (High-frequency trading, 2011). Tyto strategie, které mohou kombinovat pasivní a aktivní metody tvorby trhu, si klade za cíl profitovat z rozdílu mezi nákupní a prodejní cenou.

Arbitrážní techniky si klade za cíl profitovat z cenových rozdílů mezi trhy nebo mezi souvisejícími finančními produkty (Rise of the machines: Algorithmic trading in the foreign exchange market, 2014). Některé běžné arbitrážní strategie zahrnují statistickou arbitráž, indexovou arbitráž a fúzní arbitráž.

Strategie založené na momentu a trendech (momentum-based and trend-following strategies) předpokládají, že finanční produkty s pozitivními cenovými trendy budou i nadále takto pokračovat i v budoucnosti. Tyto metody používají algoritmické obchodníky k rozpoznávání trendů a momentu na základě tržních dat a k adekvátnímu obchodování (Momentum strategies: Evidence from the Pacific Basin stock markets, 2012).

Koncepce převrácení průměru (mean reversion) spočívá v tom, že hodnoty finančních aktiv se nakonec vrátí k jejich historickým průměrům. Algoritmický obchodníci, kteří sledují odchylky

cen od historických normálních hodnot a vstupují do obchodů, když očekávají převrácení, používají tyto strategie (Avellaneda, 2010).

Strategie založené na zprávách (news-based strategies) hledají informace ovlivňující trhy analýzou novinových článků, sociálních médií a jiných zdrojů textových dat pomocí technik zpracování přirozeného jazyka (natural language processing, NLP) (Automated news reading: Stock price prediction based on financial news using context-capturing features, 2013). Poté na základě toho, jak zprávy ovlivňují finanční produkty nebo jak se lidé cítí, provádí tyto přístupy obchody.

Strategie provedení obchodu (execution strategies) si klade za cíl maximalizovat provedení velkých objednávek a minimalizovat vliv na trh a slippage. Mezi běžné strategie provedení obchodu patří implementační deficit, vážený průměr čas u transakce (time-weighted average price, TWAP) a vážený průměr objemu obchodu (volume-weighted average pricing, VWAP) (Almgren, 2005).

3.1.3.1 VYSOKO FREKVENČNÍ OBCHODOVÁNÍ (HFT)

Konkrétní podskupina algoritmického obchodování známá jako obchodování s vysokou frekvencí (HFT) využívá sofistikované infrastruktury a technologie k rychlému provádění transakcí – někdy v milisekundách nebo mikrosekundách (Aldridge, 2010). Pro využití krátkodobých tržních neefektivit, nerovnováh likvidity a cenových rozdílů HFT strategie zahrnují umisťování velkého počtu objednávek a okamžité zrušení nebo změnu těchto objednávek v reakci na změny tržních podmínek. Pro HFT jsou nezbytné co-location, rychlé datové toky s nízkou latencí a systémy s velmi rychlým provedením objednávek (Low-latency trading, 2013).

3.1.3.2 ALGORITMICKÉ OBCHODOVÁNÍ

Algoritmické obchodování je proces využívání předem naprogramovaného software k automatizaci obchodních aktivit, včetně generování a odesílání obchodních příkazů, sledování tržních podmínek a provádění obchodů na základě předem definovaných kritérií. (Ernest, 2021)Algoritmické obchodování je často využíváno institucionálními investory a hedge fondy ke

zlepšení rychlosti vykonávání obchodů a snížení transakčních nákladů. Používá se také pro širokou škálu investičních strategií, včetně vysokofrekvenčního obchodování (HFT) a statistické arbitráže. (Ernest, 2017)

3.1.3.3 UMĚLÁ INTELIGENCE A STROJOVÉ UČENÍ

Strojové učení (ML) a umělá inteligence (AI) jsou stále častěji využívány v algoritmických obchodních systémech, zejména díky schopnosti těchto technologií spravovat a analyzovat obrovské množství dat v reálném čase (Machine learning in finance: The case of deep learning for option pricing, 2020). Díky tomu, že algoritmy ML a AI automaticky učí se z a přizpůsobují se měnícím se tržním podmínkám, je možné vytvářet dynamické obchodní modely, které se neustále vyvíjejí a optimalizují (Dynamic neural networks in high-frequency trading, 2016). Existuje několik populárních technik strojového učení používaných v algoritmickém obchodování, včetně neuronových sítí, podpůrných vektorových strojů a posilování učení (Universal features of price formation in financial markets: Perspectives from deep learning, 2019).

3.2 VÝHODY A NEVÝHODY AUTOMATIZOVANÉHO OBCHODOVÁNÍ

3.2.1 VÝHODY

3.2.1.1 RYCHLOST A EFEKTIVITA

Algoritmické obchodní systémy mohou díky schopnosti sbírat a analyzovat tržní data nezískatelnou rychlostí rychle identifikovat obchodní příležitosti a provádět objednávky (Algorithmic trading and the market for liquidity, 2013). Díky své rychlostní výhodě mohou obchodníci využívat arbitrážní příležitosti a tranzitorní tržní neefektivity a zvyšovat tak své zisky. Algoritmy mohou také analyzovat data z několika zdrojů současně, což obchodníkům poskytuje celkové porozumění tržní situaci (Gomber, 2011).

Protože automatizované obchodní systémy mohou provádět transakce bez potřeby lidského vstupu, efektivita algoritmického obchodování se také rozšiřuje na snížení manuálního zapojení (The science of algorithmic trading and portfolio management, 2013). Algoritmické obchodování může zlepšit celý obchodní proces a zvýšit šance na úspěšné provedení obchodu tím, že sníží čas potřebný k umístění a provedení objednávek.

3.2.1.2 SNÍŽENÉ RIZIKO LIDSKÉ CHYBY

Automatizované obchodní systémy mohou provádět transakce přesně, snižují tak možnost lidských chyb a zajišťují konzistenci v obchodních rozhodnutích (The science of algorithmic trading and portfolio management, 2013). Když jsou emoce odstraněny z rozhodovacího procesu, jsou produkovány obchodní techniky, které jsou logičtější a objektivnější. Ovlivnění psychologickými zkresleními, které mohou negativně ovlivnit výkon obchodování, lze minimalizovat programováním algoritmických obchodních modelů k dodržování přísných obchodních pravidel (Murphy, 2012).

Další výhodou algoritmických obchodních systémů je jejich schopnost neustále monitorovat tržní podmínky a v reálném čase měnit nastavení obchodu, což minimalizuje riziko chyb způsobených únavou nebo přetížením informacemi (Rise of the machines: Algorithmic trading in the foreign exchange market, 2014). Tato kapacita umožňuje obchodníkům udržovat svůj obchod disciplinovaný i v extrémně turbulentních nebo dynamických trzích.

3.2.2 NEVÝHODY

3.2.2.1 ZÁVISLOST NA TECHNOLOGII

Jako každý systém, i algoritmické obchodní systémy jsou závislé na technologii a mohou být náchylné k technickým poruchám nebo výpadkům (The science of algorithmic trading and portfolio management, 2013). Pokud dojde k selhání systému v klíčovém okamžiku, může to mít katastrofické následky pro obchodníka nebo investora. Proto je důležité, aby obchodníci měli plán nouzového vypnutí a zálohování dat, aby snížili riziko výpadků.

3.2.2.2 NEDOSTATEK FLEXIBILITY A PŘIZPŮSOBITELNOSTI

Přestože algoritmické obchodní systémy mohou být navrženy pro konkrétní obchodní strategie, mohou být náchylné k neúspěchu, pokud se trh nevyvíjí podle očekávání (Algorithmic trading and the market for liquidity, 2013). Algoritmy nemohou být příliš adaptabilní na neočekávané změny trhu, což může vést k neúspěšnému obchodování nebo ztrátám. Obchodníci musí mít plány pro měnící se tržní podmínky a být schopni rychle přizpůsobit své strategie.

3.2.2.3 POTENCIÁLNÍ RIZIKO SOUVISEJÍCÍ S VYSOKOFREKVENČNÍM OBCHODOVÁNÍM

Vysokofrekvenční obchodování může být náchylné k některým rizikům, včetně flash crashů a manipulace trhu (Aldridge, 2010). Pokud jsou některé transakce prováděny tak rychle, že trh nemá čas adekvátně reagovat, mohou se objevit nestabilita a chyby v tržních cenech. Obchodníci vysokofrekvenčního obchodování musí být opatrní, aby minimalizovali rizika spojená s těmito obchodními metodami.

Manipulace trhu je také rizikem spojeným s vysokofrekvenčním obchodováním. Obchodníci mohou využít rychlosti svých algoritmů k manipulaci trhu, například k vytváření umělých cenových anomálií (Low-latency trading, 2013). Tyto anomálie mohou ovlivnit chování ostatních obchodníků a vést k nevyváženosti trhu.

Dalším rizikem spojeným s vysokofrekvenčním obchodováním je technická chyba, která může způsobit ztrátu peněz nebo dokonce havárii celého systému. Tento riziko může být minimalizováno pomocí adekvátního testování a zabezpečení obchodního systému.

3.2.2.4 POTENCIÁLNÍ RIZIKO SPOJENÁ S ALGORITMICKÝM OBCHODOVÁNÍM NA ZÁKLADĚ STROJOVÉHO UČENÍ

Algoritmické obchodní systémy na základě strojového učení mohou být náchylné k riziku overfittingu, což znamená, že se systém naučí příliš dobře přizpůsobit trénovacím datům a bude mít špatnou předpovědní schopnost na nových datech (Dynamic neural networks in high-frequency trading, 2016). To může vést k nesprávným rozhodnutím a ztrátám peněz.

Dalším rizikem je špatné kvality dat, zejména pokud jsou data neúplná, zastaralá nebo zdrojové data jsou nesprávná. To může ovlivnit výkon algoritmu a vést k nesprávným obchodním rozhodnutím (Universal features of price formation in financial markets: Perspectives from deep learning, 2019).

3.2.2.5 PORUCHY SYSTÉMU A TECHNICKÁ RIZIKA

Pokud nejsou řádně řešeny, technická rizika, která jsou algoritmickým obchodním systémům vystavena, mohou zahrnovat chyby softwaru, problémy s hardwarovými zařízeními a výpadky sítě, což může způsobit významné finanční ztráty (Equilibrium fast trading, 2015). Vzhledem k složitosti a propojenosti moderních obchodních systémů existuje riziko kaskádových selhání, kdy může jediný vadný algoritmus spustit řetězec událostí, který naruší trh jako celek (Market liquidity and funding liquidity., 2009). Aby se snížila tato rizika, musí obchodníci a instituce vynakládat značné investice do infrastruktury systému, zavést přísná testovací procedury a mít záložní plány pro řešení pravděpodobných výpadků (Algorithmic trading and the market for liquidity, 2013).

3.2.3 PROMĚNNÉ OVLIVŇUJÍCÍ ALGORITMICKÉ OBCHODNÍ METODY

3.2.3.1 TRŽNÍ PODMÍNKY A ZMĚNY V ZÁKONĚ

Tržní prostředí a obchodní režim, jako jsou změny v tržní volatilitě, likviditě a korelacích aktiv, mohou mít vliv na algoritmické obchodní přístupy (What happened to the quants in August 2007? Evidence from factors and transactions data., 2007). Náhlé změny v tržním chování mohou mít vliv na úspěch nebo selhání algoritmických obchodních přístupů navržených pro určité tržní režimy. Aby se tyto fluktuace zohlednily, musí obchodníci vyvinout přizpůsobivé strategie, které se dobře osvědčí v řadě tržních situací, nebo mít několik metod pro různé tržní režimy (Volatility clustering in financial markets: Empirical facts and agent-based models., 2007).

3.2.3.2 STRUKTURA TRHU A PRÁVNÍ RÁMEC

Struktura a regulace trhu mohou mít významný vliv na účinnost algoritmických obchodních algoritmů. Účinnost konkrétních obchodních strategií může být ovlivněna, a dynamika obchodu se může změnit, když jsou změněny tržní zákony, jako jsou typy objednávek, velikosti ticků a obchodní hodiny (The diversity of high-frequency traders., 2013). Časté přijímání nových pravidel regulačními orgány v úsilí bojovat proti manipulaci trhu a dalším nekalým praktikám může potenciálně mít vliv na životaschopnost některých algoritmických strategií. Obchodníci musí sledovat vývoj regulací a upravit své postupy podle potřeby, aby zajistili dodržování předpisů a udrželi výkon (What do we know about high-frequency trading?, 2013).

3.2.3.3 SLIPPAGE A TRANSAKČNÍ NÁKLADY

Transakční náklady, jako jsou poplatky, poplatky za služby a rozdíly mezi nákupní a prodejní cenou, mohou významně ovlivnit ziskovost algoritmických obchodních strategiím (Almgren, 2005). Ziskovost může být ovlivněna vyššími transakčními náklady, zejména u vysokofrekvenčních systémů, které závisí na rychlých změnách cen. Slippage, které nastává, když jsou objednávky vyplněny za nižší c en než očekávané, může mít vliv na úspěšnost plánu (Low-

latency trading, 2013). Aby se snížily transakční náklady a slippage, obchodníci by měli pečlivě monitorovat provádění objednávek, zvažovat dopad na trh a hledat nejvýhodnější obchodní místa (The science of algorithmic trading and portfolio management, 2013).

3.2.3.4 KVALITA A DOSTUPNOST DAT

Kvalita a dostupnost dat jsou pro vývoj a úspěch algoritmických obchodních systémů klíčové. Nesprávná, neúplná nebo zastaralá data mohou vést k nesprávným modelům a špatným obchodním rozhodnutím (Universal features of price formation in financial markets: Perspectives from deep learning, 2019). Kromě toho mohou časově přesné a kvalitní data poskytnout konkurenční výhodu tím, že umožní rychlejší a přesnější rozhodování (High-frequency trading. Business & Information Systems Engineering, 2011). Obchodníci by měli investovat do spolehlivých zdrojů dat a vyvinout metody pro hodnocení a čištění svých dat, aby získali z jejich strategií maximální výkon.

3.2.3.5 VĚCI, KTERÉ JE TŘEBA ZVÁŽIT PŘI ŘÍZENÍ RIZIK

Efektivní řízení rizik je nezbytné pro dlouhodobý úspěch algoritmických obchodních systémů. Operační, tržní a kreditní rizika musí být efektivně řízena, protože všechna mohou ovlivnit úspěšnost strategie (Classification-based financial markets prediction using deep neural networks, 2020). Obchodníci by měli vyvinout rizikové modely s vyváženým poměrem rizik a odměn, které zohledňují specifická rizika spojená s jejich přístupy. Řízení rizik by také mělo zahrnovat stresové testování a analýzu scénářů, aby se zajistilo, že plány mohou odolat nepříznivým tržním podmínkám (Murphy, 2012).

3.2.3.6 FLEXIBILITA A SLOŽITOST ALGORITMICKÝCH TECHNIK

Složitost a flexibilita algoritmických obchodních systémů v různých tržních podmínkách mohou mít vliv na jejich výkon a odolnost. Příliš složité strategie se mohou potýkat s obtížemi

přizpůsobit se měnícím se tržním podmínkám a mohou být náchylnější k přeučení (overfitting) (Bailey, 2014). Na druhé straně mohou být jednoduché strategie nedostatečně komplexní na využití sofistikovanějších obchodních příležitostí. Pro vývoj efektivních algoritmických obchodních systémů, které mohou odolat volatilitě finančních trhů, je nutné najít rovnováhu mezi složitostí a přizpůsobivostí (The adaptive markets hypothesis: Market efficiency from an evolutionary perspective, 2004).

3.3 ALGORITMICKÉ OBCHODNÍ METODY A TECHNICKÉ NÁSTROJE ANALÝZY

3.3.1 FORMACE SVÍČKOVÝCH GRAFŮ

V algoritmickém obchodování jsou formace svíčkových grafů oblíbenou metodou technické analýzy pro hledání potenciálních obchodních příležitostí založených na cenových vzorech (Nison, 1991). Každý svíčkový graf zobrazuje hodnoty otevření, uzavření, nejvyšší a nejnižší ceny jako reprezentaci změn cen v určitém časovém období. Formace svíčkových grafů, jako jsou hammer, doji, engulfing a harami, mezi jinými, mohou poskytovat důležité analýzy tržních emocí a potenciálních budoucích změn cen (Morris, 2012). Tyto vzory se často používají v algoritmických obchodních algoritmech pro generování nákupních a prodejních signálů na základě minulého výkonu a předvídatelnosti (Candlestick technical trading strategies: Can they create value for investors?, 2006).

3.3.2 KLOUZAVÉ PRŮMĚRY

V algoritmickém obchodování se často používají klouzavé průměry (MAs) k vyhlazení cenových dat a hledání vzorů (Kirkpatrick, 2010). Klouzavé průměry pomáhají odstranit šum a zdůraznit základní trendy odhadem průměrné ceny aktiv na určité období. Klouzavé průměry existují v různých formách, každá s konkrétními vlastnostmi a použitím, jako jsou jednoduché klouzavé průměry (SMA), exponenciální klouzavé průměry (EMA) a vážené klouzavé průměry (WMA) (Hull, 2013).

Při křížení dvou klouzavých průměrů s různými časovými obdobími se generují potenciální nákupní nebo prodejní signály. Toto se nazývá křížení klouzavého průměru. Například křížení krátkodobého MA nad dlouhodobým MA může signalizovat býčí signál, zatímco křížení krátkodobého MA pod dlouhodobým MA může signalizovat medvědí signál (Brown, 2008). Křížení klouzavých průměrů se často používají v algoritmických obchodních systémech k odhalení změn trendu a generování obchodních signálů na základě minulého výkonu (Murphy, 2012).

3.3.3 TRENDOVÉ ČÁRY A KANÁLY

V technické analýze jsou trendové čáry a kanály klíčovými nástroji pro odhalování a hodnocení tržních trendů (Kirkpatrick, 2010). Trendové čáry se vytvářejí spojením dvou nebo více významných cenových bodů, jako jsou vrcholy nebo dna, aby ukázaly směr a sklon trendu (Pring, 2002). Kanály vznikají paralelním spojením cenových výkyvů a představují úrovně podpory a odporu pro aktivum (Schwager, 1996).

Trendové čáry a kanály se často používají v algoritmických obchodních systémech k hledání možných vstupních a výstupních příležitostí na základě minulých trendů a cenových vzorců (Is technical analysis in the foreign exchange market profitable? A genetic programming approach., 1997). Použitím těchto nástrojů mohou algoritmický obchodníci vytvářet strategie, které profitují na obratech trendu nebo následují trend, aby zlepšili svůj obchodní úspěch (Murphy, 2012).

3.3.4 ÚROVNĚ PODPORY A ODPORU

Technická analýza silně spoléhá na koncepty úrovní podpory a odporu, což jsou úrovně cen, přes které se cena aktivu historicky obtížně zvyšovala (odpor) nebo snižovala (podpora) (Kirkpatrick, 2010). Tyto úrovně se používají k vytváření ztrátových zastavení, ziskových cílů a pravděpodobných vstupních a výstupních bodů pro obchody (Murphy, 2012). Úrovně podpory a odporu se často používají v algoritmických obchodních systémech k poskytování obchodních signálů na základě minulých cenových vzorců a psychologie trhu (Support and resistance: Trading by reading a market, 2015).

3.3.5 GRAFICKÉ TVARY

Grafické tvary jsou vizuální reprezentace pohybů cen, které se objevují jako rozpoznatelné tvary nebo formace, jako jsou trojúhelníky, hlava a ramena, dvojité vrcholy a dvojitá dna (Bulkowski, 2005). Tyto formace odhalují emoce trhu a pomáhají předpovídat, zda trend bude pokračovat nebo se obrátí (Is technical analysis in the foreign exchange market profitable? A genetic programming approach., 1997). Grafické tvary se často používají v algoritmických obchodních systémech ke generování obchodních signálů, protože nabízejí systematický způsob, jak najít obchodní příležitosti na základě minulého cenového vývoje (Kirkpatrick, 2010).

3.3.6 TECHNICKÉ INDIKÁTORY

Technické indikátory jsou matematické výpočty provedené na minulých cenových a objemových datech, které pomáhají obchodníkům identifikovat trendy, moment a pravděpodobné obraty trhu (Achelis, 2000). Když se používají spolu s jinými nástroji technické analýzy, podporují obchodní rozhodnutí a generují obchodní signály pro algoritmické obchodní techniky (Kaufman, 2013).

3.3.6.1 INDIKÁTORY MOMENTU (NAPŘ. RSI, STOCHASTICKÝ OSCILÁTOR)

Technické nástroje nazývané indikátory momentu pomáhají obchodníkům identifikovat překoupené nebo přeprodané pozice, stejně jako možné obraty trendu tím, že měří rychlost a sílu cenových pohybů (Pring, 2002). Relativní síla index (RSI) a stochastický oscilátor jsou dva často používané indikátory momentu.

RSI byl vytvořen J. Wellesem Wilderem a pomáhá obchodníkům rozpoznat překoupené nebo přeprodané situace tím, že porovnává velikost předchozích zisků s nedávnými ztrátami (New concepts in technical trading systems., 1978). Stochastický oscilátor byl vytvořen Georgem C. Lanem a vyhodnocuje uzávěrovou cenu vzhledem k cenovému rozpětí po určitém čase a také signalizuje překoupené nebo přeprodané situace (Lane's stochastic., 1984). Tyto indikátory

momentu jsou často používány algoritmickými obchodními systémy k vytváření obchodních doporučení na základě minulých cenových pohybů a momentu trhu.

3.3.6.2 INDIKÁTORY TRENDU (NAPŘ. MACD, ADX)

Technické indikátory známé jako "indikátory trendu" pomáhají obchodníkům rozpoznat a hodnotit sílu stávajících trendů na finančních trzích (Kaufman, 2013). Moving Average Convergence Divergence (MACD) a Average Directional Index (ADX) jsou dva často používané indikátory trendu.

MACD Geralda Appela, který je založen na momentu, analyzuje vztah mezi dvěma klouzavými průměry, obvykle 12denním a 26denním exponenciálním klouzavým průměrem (Appel, 1988). Zatímco signální linka (často 9denní EMA) se používá k poskytování obchodních signálů na základě křížení, MACD linka se generuje odečtením delšího klouzavého průměru od kratšího (Kirkpatrick, 2010).

ADX, který byl vytvořen J. Wellesem Wilderem, je indikátorem síly trendu, který kvantifikuje, jak silně se aktivum mění bez zohlednění směru trendu (New concepts in technical trading systems., 1978). Pozitivní směrový indikátor (+DI) a negativní směrový indikátor (-DI), které se používají k kvantifikaci směrového pohybu, jsou dalšími indikátory, ze kterých se ADX odvodí (Pring, 2002). Tyto indikátory trendu se často používají v algoritmických obchodních systémech k vytváření obchodních signálů na základě intenzity a směru tržních trendů.

3.3.6.3 INDIKÁTORY VOLATILIT

Technické indikátory známé jako indikátory volatilit slouží obchodníkům k detekci časů vysoké nebo nízké volatility měřením cenových změn na finančních trzích (Kaufman, 2013). Average True Range (ATR) a Bollinger Bands jsou dva často používané indikátory volatility.

Bollinger Bands, které vytvořil John Bollinger, se skládají z klouzavého průměru (často 20denní SMA) a dvou pásem standardních odchylek nad a pod klouzavým průměrem (Bollinger, 2001). Na základě volatility na trhu se tato pásma rozšiřují a zužují, poskytují obchodníkům

informace o možných překoupených nebo přeprodaných pozicích a o pravděpodobných obratech cen.

Dalším indikátorem volatility je ATR, který vytvořil J. Welles Wilder a vypočítává typický rozsah pohybu cen během určitého časového období (New concepts in technical trading systems., 1978). ATR se používá k hodnocení volatility na trhu a k určování úrovní stop-loss, což pomáhá obchodníkům úspěšněji řídit riziko. Tyto indikátory volatility jsou často používány v algoritmických obchodních technikách k řízení rizika a k vytváření obchodních signálů na základě volatility na trhu.

3.3.6.4 INDIKÁTORY OBJEMU (NAPŘ. INDEX TOKU PENĚZ (MFI) A ON-BALANCE VOLUME (OBV))

Technické indikátory známé jako "indikátory objemu" měří sílu změn cen na základě objemu obchodů (Achelis, 2000). On-Balance Volume (OBV) a Index toku peněz jsou dva populární objemové ukazatele (MFI).

OBV, který vytvořil Joe Granville, je kumulativním objemovým indikátorem, který přičítá objem v dny s růstem a odčítá objem v dny s poklesem, aby zobrazil kupní a prodejní tlak (Granville, 1963). OBV může pomoci obchodníkům v identifikaci pravděpodobných obratů trendu tím, že ukazuje rozdíly mezi cenou a objemem.

MFI, indikátor objemu založený na momentum, který vytvořili Gene Quong a Avrum Soudack, sleduje pohyb peněz do a z aktiva během určitého časového období MFI je podobné RSI, ale bere také v úvahu informace o objemu a cenách, aby pomohl obchodníkům identifikovat překoupené nebo přeprodané pozice a pravděpodobné obraty trendu. Tyto objemové ukazatele jsou často používány v algoritmických obchodních algoritmech k vytváření obchodních signálů na základě tržního sentimentu a dynamiky poptávky a nabídky (Kaufman, 2013).

3.4 METRIKY PRO HODNOCENÍ ALGORITMICKÝCH OBCHODNÍCH STRATEGIÍ

Porozumění úspěchu algoritmických obchodních technik a identifikace příležitostí pro rozvoj vyžadují pravidelné hodnocení výkonu. Účinnost obchodních strategií se hodnotí pomocí

různých kritérií, včetně absolutních metrik výkonu, metrik výkonu upravených na riziko a dalších specializovaných měření (Bailey, 2014).

3.4.1 OBJEKTIVNÍ UKAZATELE VÝKONU

Bez zohlednění míry rizika spojeného se strategií se absolutní metriky výkonu zaměřují na čistý výkon obchodní metody.

3.4.1.1 CELKOVÝ ZISK

Celkový výnos, prezentovaný jako procento počáteční investice, je měřením celkového zisku nebo ztráty, kterou obchodní strategie vytvořila během určitého období. Vypočte se dělením počáteční investice celkovým ziskem z kapitálových zisků, dividend a úroků (Bodie, 2014).

3.4.2 METRIKY VÝKONNOSTI, KTERÉ ZOHLEDŇUJÍ RIZIKO

Metriky výkonnosti upravené o riziko zohledňují riziko obchodní strategie a poskytují podrobnější hodnocení jejího výkonu. Tyto metriky umožňují obchodníkům hodnotit strategie s různými profil rizika a identifikovat ty, které poskytují nejlepší poměr riziko/výnos.

3.4.2.1 SHARPE RATIO

William Sharpe vytvořil Sharpeho poměr, který porovnává nadbytečný výnos investiční strategie (tj. výnos nad bezrizikovou mírou) s jejím stupněm rizika, jak je určeno standardní odchylkou výnosů (Mutual fund performance, 1966). Vyšší Sharpeho poměr ukazuje na vylepšený výnos upravený o riziko pro danou strategii.

3.4.2.2 SORTINO RATIO

Sortino ratio je variantou Sharpeho poměru, kterou vytvořil Frank Sortino a bere v úvahu pouze downside riziko, jak je ukázáno standardní odchylkou záporných výnosů (Performance

measurement in a downside risk framework, 1994). Tato statistika je velmi užitečná pro strategie s asymetrickými distribucemi výnosů nebo ty, které kladou důraz na ochranu kapitálu.

3.4.2.3 CALMAR RATIO

Calmar ratio, někdy nazýván Sterling ratio, hodnotí výkonnost obchodní strategie upravené o riziko porovnáním jejího ročního výnosu s největším propadem (Learning the art of the science of trading: The Sterling ratio, 1991). Calmar ratio je užitečný indikátor pro hodnocení odolnosti strategie v obtížných tržních podmínkách, protože vyšší hodnota ukazuje na lepší výkon vzhledem k nejhorší ztrátě pro danou strategii.

3.4.2.4 TREYNOR RATIO

Treynor ratio, který vytvořil Jack Treynor, hodnotí výkonnost obchodní strategie upravené o riziko porovnáním jejího nadbytečného výnosu s množstvím systematického rizika ukazovaného beta dané strategie (How to rate management of investment funds., 1965). Treynor ratio je užitečný indikátor pro hodnocení výkonu diverzifikovaných portfolií, protože vyšší hodnota naznačuje větší výnos na jednotku systematického rizika.

3.4.2.5 OMEGA RATIO

Omega ratio, který vytvořili Con Keating a William Shadwick, hodnotí upravený výkonnostní ukazatele rizika obchodní strategie porovnáváním pravděpodobnostně vážených zisků a ztrát vůči určitému cílovému výnosu (A universal performance measure, 2002). Omega ratio je užitečným ukazatelem pro hodnocení úspěšnosti strategií s ne normálními rozděleními výnosů, protože vyšší hodnota značí příznivější vyvážení mezi potenciálními zisky a ztrátami.

3.4.3 SLEDOVÁNÍ VÝKONNOSTI V PRŮBĚHU ČASU

Porozumění odolnosti obchodní strategie a schopnosti reagovat na změny na trhu vyžaduje vyhodnocení její výkonnosti v průběhu času. K tomu lze použít několik přístupů, včetně analýzy koulejících oken, optimalizace procházení a testování mimo výběrový soubor.

3.4.3.1 ANALÝZA KOULEJÍCÍCH OKEN

Technika nazývaná analýza koulejících oken se používá k posouzení účinnosti obchodní strategie během několika překrývajících se časových období. Zahrnuje segmentaci historických dat do několika překrývajících se oken, z nichž každé reprezentuje odlišné časové období (například tři měsíce, rok) (Pardo, 2008). Vypočte se statistika výkonnosti každého okna a porovná se, přičemž obchodní strategie se hodnotí v každém okně. Tato analýza ukazuje, jak konzistentně a stabilně se v průběhu času udržela výkonnost strategie.

3.4.3.2 OPTIMALIZACE PROCHÁZENÍ VPŘED

Přístup optimalizace procházení vpřed testuje odolnost a přizpůsobivost obchodní strategie tím, že iterativně reoptimalizuje parametry strategie a posuzuje výkonnost strategie na nových, mimo výběrový soubor dat (Pardo, 2008). Parametry přístupu se optimalizují pomocí určité části historických dat, nazývané "výběrová data", která byla rozdělena do různých segmentů. Poté se posoudí výkonnost optimalizované strategie, když se použije na další segment dat, nazývaný také "mimo výběrový soubor" dat. Opakované opakování tohoto postupu pro každou část vytváří sadu výkonnostních měření mimo výběrový soubor, které lze použít k posouzení toho, jak dobře lze strategii přizpůsobit se měnícím se podmínkám na trhu.

3.4.3.3 TESTOVÁNÍ MIMO VÝBĚROVÝ SOUBOR

Testování mimo výběrový soubor je technika validace výkonnosti obchodní strategie pomocí datové sady, která nebyla použita během procesu optimalizace (Bailey, 2014). Tato metoda poskytuje přesnější hodnocení výkonnosti strategie v reálných obchodních podmínkách a

snižuje riziko přizpůsobení strategie historickým datům. Historická data jsou často rozdělena do dvou nebo více částí, z nichž jedna je použita pro validaci a ostatní pro optimalizaci (in-sample data) (out-of-sample data). Účinnost obchodní techniky je pak hodnocena pomocí out-of-sample dat za účelem určení, zda ji lze použít v nových, nezkoumaných tržních podmínkách.

3.4.4 SROVNÁNÍ S BENCHMARKY

Vyhodnocení výkonnosti obchodní strategie v porovnání s jinými investicemi nebo tržními indexy je nezbytné provést pomocí srovnání s benchmarky. Investoři mohou posoudit účinnost strategie a zjistit, zda dosahuje svých investičních cílů, porovnáním jejího výkonu s relevantními benchmarky. Srovnání s benchmarky může být provedeno různými způsoby, například pomocí tržních indexů, porovnání s konkurenční skupinou nebo srovnání s bezrizikovou mírou výnosu.

3.4.4.1 INDEXY TRHU

Tržní indexy slouží jako standardní metrika pro hodnocení, jak dobře obchodní strategie fungovala v porovnání s celkovým trhem nebo konkrétním odvětvím. Indexy jako S&P 500, NASDAQ Composite a Dow Jones Industrial Average měří výkon celkového trhu nebo tržního segmentu a slouží jako benchmark pro posouzení účinnosti strategie (Bodie, 2014). Investoři mohou porovnáním výnosů strategie s výnosy relevantního indexu zjistit, zda strategie překonává nebo zaostává za trhem.

3.4.4.2 SROVNÁNÍ MEZI KONKURENTY

V srovnání mezi konkurenty se porovnává výkon obchodní strategie s výkonem skupiny jiných obchodních strategií nebo investičních produktů, které jsou s ní srovnatelné, jako jsou podílové fondy, hedgeové fondy nebo jiné algoritmické obchodní techniky. Tento způsob umožňuje investorům posoudit relativní výkon strategie v rámci daného tržního segmentu nebo investičního stylu (Brown, 2008). Srovnání s konkurenční skupinou lze použít k určení, zda obchodní strategie překonává jiné podobné strategie nebo zda je její výkon v podstatě průměrný v porovnání s normami odvětví.

3.4.4.3 SROVNÁNÍ S BEZRIZIKOVOU MÍROU VÝNOSU

Benchmarkem pro měření upraveného výkonu obchodní strategie vzhledem k riziku je výnos krátkodobých státních dluhopisů, jako jsou Treasury bills, které slouží jako proxy pro bezrizikovou míru (Sharpe, 1994). Bezriziková míra funguje jako benchmark pro minimální očekávaný výnos, který by měl investor získat při investování do bezrizikového aktiva. Porovnáním nadbytečného výnosu (tj. výnosu nad bezrizikovou mírou) s mírou rizika mohou investoři určit, zda daná strategie vytváří odpovídající výnosy vzhledem k přijatému riziku.

3.4.5 ANALYTICKÉ POSTUPY A OVĚŘENÍ MODELU

Pro analýzu statistické významnosti výkonnosti obchodní strategie a odhalení potenciálních zkreslení a overfittingu je nutné použít statistické testy a postupy ověřování modelu. T-test, F-test, informační poměr a Jensenův alfa jsou některé statistické testy, které lze použít k posouzení úspěšnosti obchodní strategie.

3.4.5.1 T-TEST

T-test je statistická analýza používaná k zjištění, zda průměrný výnos obchodní strategie se významně liší od daného benchmarku, jako je bezriziková míra nebo tržní index (Student, 1908). T-test odvodí t-statistiku, která se porovná s kritickou hodnotou z t-rozdělení, aby se posoudilo, zda průměrný výnos strategie významně od benchmarku odchyluje při určité úrovni důvěry. Pokud je t-statistika významná, je pravděpodobné, že úspěch strategie se nedostavil pouze náhodou.

3.4.5.2 F-TEST

F-test je statistický test prováděný při porovnávání rozptylů dvou nebo více obchodních strategií nebo investičních portfolií (Fisher, 1925). Pokud se rozptyly strategií významně liší při určité úrovni důvěry, F-test odvodí F-statistiku, která se porovná s kritickou hodnotou z F-

rozdělení. Významná F-statistika naznačuje, že úrovně rizika nebo variability výnosů strategií jsou významně odlišné.

3.4.5.3 INFORMAČNÍ POMĚR

Informační poměr je rizikem upravená výkonnostní metrika, která porovnává nadbytečný výnos obchodní strategie s tracking errorem, což je standardní odchylka rozdílu mezi výnosy strategie a benchmarkovými výnosy (The information ratio, 1998). Vyšší informační poměr ukazuje lepší rizikem upravenou výkonnost v porovnání s benchmarkem. Informační poměr lze použít jako základ pro testování hypotéz; významný informační poměr naznačuje, že nadbytečný výnos strategie se nedostavil pouze náhodou.

3.4.5.4 BETA JENSENOVA ALFA

Jensenova alfa je ukazatelem výkonnosti, který vypočítá anomální výnos obchodní strategie po kompenzaci systematického rizika, které je vyjádřeno beta koeficientem (The performance of mutual funds in the period 1945-1964, 1968). Pozitivní hodnota Jensenovy alfy znamená, že strategie překonává očekávání vzhledem k jejímu riziku, zatímco negativní alfa značí podprůměrný výkon. Pro posouzení statistické významnosti anomálního výnosu obchodní strategie lze Jensenovu alfu testovat pomocí t-testu; vysoká hodnota alfy znamená, že výkon strategie pravděpodobně není náhodným jevem.

4 PRAKTICKÁ ČÁST

4.1 NÁVRH A PŘÍSTUP K VÝZKUMU

Tato kapitola popisuje návrh a přístup k výzkumu použité v této práci za účelem navržení, implementace a testování algoritmu pro intradenní momentum obchodování s akciemi. Poskytuje komplexní přehled o účelu výzkumu, otázkách, hypotézách, metodologii a omezeních.

4.1.1 ÚČEL VÝZKUMU

Hlavním účelem tohoto výzkumu je vyvinout účinný algoritmus pro intradenní momentum obchodování s akciemi, který identifikuje nejvíce obchodované akcie v daný den a generuje nákupní a prodejní signály na základě jednoduchých formací candlesticků, vybraných pravidel, ukazatelů a oscilátorů. Kromě toho si výzkum klade za cíl navrhnout a implementovat desktopovou aplikaci s grafickým rozhraním, integrovanou s obchodní platformou Interactive Brokers, pro správu, řízení, úpravu a nastavení algoritmu v reálném čase.

4.1.2 VÝZKUMNÉ OTÁZKY

Následující výzkumné otázky vedou studii:

1. Jak lze navrhnout algoritmus pro detekci nejvíce obchodovaných akcií v daný den a identifikovat jednoduché formace candlesticků pro nákupní a prodejní signály?
2. Která pravidla, ukazatele a oscilátory jsou nejúčinnější pro generování nákupních a prodejních signálů při intradenním momentum obchodování s akciemi na základě historických dat?
3. Jaká je výkonnost vyvinutého algoritmu v reálném obchodním prostředí a jak se porovnává s etablovanými benchmarky pomocí Sharpeho poměru?

4.1.3 VÝZKUMNÉ HYPOTÉZY

Na základě výzkumných otázek jsou formulovány následující hypotézy:

H1: Navržený algoritmus dokáže účinně identifikovat nejvíce obchodované akcie v daný den a generovat nákupní a prodejní signály pomocí jednoduchých formací candlesticků.

H2: Vybraná pravidla, ukazatele a oscilátory mohou zlepšit účinnost algoritmu při intradenním momentum obchodování na základě historických dat.

H3: Navržená desktopová aplikace s GUI může efektivně spravovat, ovládat, upravovat a nastavovat algoritmus v reálném čase, když je integrována s obchodní platformou Interactive Brokers.

H4: Vyvinutý algoritmus prokazuje akceptovatelnou výkonnost v reálném prostředí obchodování, měřenou Sharpeovým poměrem.

4.1.4 VÝZKUMNÁ METODOLOGIE

Metodologie výzkumu pro tuto práci se skládá z následujících etap:

Teoretické základy: Přezkum relevantní literatury o obchodování s akciemi, algoritmickém obchodování, hodnocení výkonnosti algoritmů a metodách, ukazatelích, oscilátorech a programovacích nástrojích použitých v této práci.

Návrh algoritmu: Vývoj algoritmu, který detekuje jednoduché formace svíček identifikací nejvíce obchodovaných akcií na daný den.

Backtesting a optimalizace: Identifikace nejefektivnějších pravidel, ukazatelů a oscilátorů pro nákup a prodej na základě backtestingu s použitím historických dat a statistické analýzy.

Vývoj aplikace: Návrh a implementace desktopové aplikace s GUI propojené s obchodní platformou Interactive Brokers pro řízení a ovládání algoritmu v reálném čase.

Testování v reálném prostředí a hodnocení výkonnosti: Implementace a testování konečného algoritmu v reálném prostředí obchodování pomocí navržené desktopové aplikace, následované hodnocením výkonnosti pomocí Sharpeova poměru.

4.1.5 OMEZENÍ VÝZKUMU

Výzkum čelí několika omezením, jako jsou:

Závislost na historických datech pro backtesting a optimalizaci, které nemusí přesně předpovídat budoucí tržní podmínky.

Potenciální vliv změn tržního režimu a externích faktorů na výkon algoritmu.

Omezený rozsah studie, zaměřený na intradenní momentum obchodování a jednoduché formace svíček, které nemusí zahrnovat celé spektrum možných obchodních strategií a technik.

Integrace s obchodní platformou Interactive Brokers, která může omezit zobecnitelnost zjištění na jiné obchodní platformy.

Použití Sharpeova poměru jako hlavního ukazatele výkonnosti algoritmu, což nemusí zahrnovat všechny aspekty jeho rizikově upravené výkonnosti.

Navzdory těmto omezením má výzkum za cíl poskytnout cenné informace o návrhu, implementaci a vyhodnocování výkonnosti algoritmu pro intradenní momentum obchodování. Tento výzkum přispívá k pochopení strategií algoritmického obchodování a nabízí praktická řešení pro řízení algoritmu v reálném čase pomocí GUI desktopové aplikace integrované s obchodní platformou Interactive Brokers.

4.2 METODY SBĚRU A ANALÝZY DAT

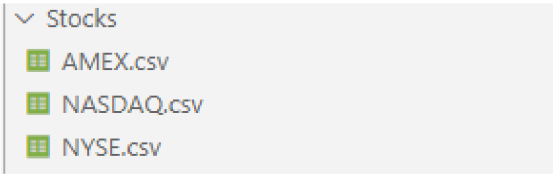
Úspěch jakékoli algoritmické obchodní strategie výrazně závisí na kvalitě a přesnosti použitých dat při vývoji a implementaci. Proto v této části popíšeme metody sběru a analýzy dat, které budou použity při návrhu, implementaci a testování našeho algoritmu pro intradenní momentum obchodování s akciemi.

4.2.1 ZDROJE DAT

Zdrojem dat pro náš algoritmus bude AlphaVantage, poskytovatel kvalitních finančních tržních dat nabízejících v reálném čase a historická data pro akcie, forex a kryptoměny.

AlphaVantage poskytuje rozsáhlou sadu API, která umožňuje přístup k různým finančním datovým bodům, jako jsou cena, objem a technické ukazatele.

Taktéž, využíváme zdrojů od burzy NASDAQ pro kumulativní data o všech společnostech aktuálně zapsaných na burzách NASDAQ, NYSE a AMEX. Zde se nachází data: Symbol, Název, Poslední prodej, Netová změna, % Změna, Tržní kapitalizace, Země, Rok IPO, Objem, Sektor, Průmysl - (Symbol,Name,Last Sale,Net Change,% Change,Market Cap,Country,IPO Year,Volume,Sector,Industry)



OBRÁZEK 1 - CSV SOUBORY V PROGRAMU

4.2.2 POSTUPY SBĚRU DAT

Pro sběr dat potřebných pro náš algoritmus použijeme API AlphaVantage ke stažení intradenních cenových dat akcií nejvíce obchodovaných na burze. Omezíme sběr dat na posledních 730 dní intradenního obchodování, abychom zajistili použití nedávných a relevantních dat pro vývoj našeho algoritmu.

4.2.3 TECHNIKY ANALÝZY DAT

Pro analýzu shromážděných dat použijeme kombinaci statistických a technických analýz. Začneme analýzou základních statistik cenových dat akcií, jako jsou průměr, směrodatná odchylka a sklon, abychom získali lepší představu o distribuci dat.

Poté použijeme technické analýzy, jako je identifikace svíčkových formací, klouzavých průměrů a dalších technických ukazatelů pro detekci signálů nákupu a prodeje. V následujícím kroku použijeme technickou analýzu, například identifikaci svíčkových formací, klouzavých průměrů a dalších technických ukazatelů k detekci signálů nákupu a prodeje. Tyto nástroje nám

pomohou identifikovat obchodní příležitosti na základě historických dat, což nám umožní optimalizovat vstupy algoritmu a zlepšit jeho výkonnost. Při analýze technických ukazatelů se zaměříme na jejich schopnost generovat signály nákupu a prodeje v souladu s momentum obchodováním. Dále budeme používat oscilátory, jako je například RSI (Relative Strength Index) a MACD (Moving Average Convergence Divergence), k potvrzení našich signálů nákupu a prodeje. Tyto nástroje umožňují identifikovat, zda se trh nachází v překoupeném nebo přeprodaném stavu, což může být užitečné pro detekci zvratů trhu a potenciálních obchodních příležitostí.

4.2.4 VALIDACE DAT

Pro zajištění přesnosti a validity dat použitých v našem algoritmu budeme provádět postupy pro validaci dat. Budeme kontrolovat chybějící data, odlehlé hodnoty a jiné anomálie v datech. Jakékoliv nesprávné body dat budou odstraněny nebo opraveny, aby se zabránilo zkreslení výkonu algoritmu.

Závěrem lze konstatovat, že metody sběru a analýzy dat popsané v této sekci budou použity pro vývoj a testování našeho algoritmu pro obchodování s akciemi s intradenním momentum. Použitím kvalitních dat od společnosti AlphaVantage a použitím přísných postupů pro analýzu dat máme za cíl navrhnout algoritmus, který bude poskytovat ziskové obchodní signály.

4.3 VÝVOJ A IMPLEMENTACE ALGORITMU

Algoritmické obchodování se stalo nezbytnou součástí moderního finančního světa, umožňující obchodníkům provádět velké obchody s vysokou rychlostí a efektivitou. Cílem této práce je navrhnout, implementovat a otestovat algoritmus pro obchodování s akciemi s intradenním momentum, s primárním úkolem identifikovat nejvíce obchodované akcie na burze v určitý den a detekovat signály k nákupu a prodeji pomocí nově vznikajících jednoduchých svíčkových formací a pravidel, ukazatelů a oscilátorů vybraných námi.

4.3.1 VÝVOJ ALGORITMU

Vývoj algoritmu se skládá ze tří částí. V první části představíme problém a popíšeme důležité pojmy související s obchodováním s akciemi, algoritmickým obchodováním, hodnocením výkonu algoritmu a metodami, ukazateli, oscilátory a programovacími nástroji používanými v této oblasti. Ve druhé části navrhneme algoritmus, který bude detekovat jednoduché svíčkové formace tím, že identifikuje vyšší maxima nebo nižší minima u nejvíce obchodovaných akcií v určitý den. V třetí části budou identifikovány nejúčinnější pravidla, ukazatele a oscilátory pro nákup a prodej na základě backtestování pomocí historických dat prostřednictvím implementovaného algoritmu pro detekci formací z druhé části.

Pro návrh algoritmu, který dokáže efektivně identifikovat jednoduché svíčkové formace, použijeme programovací jazyk Python, který se stal standardním nástrojem při vývoji algoritmických obchodních strategií. Python je známý pro svoji jednoduchost použití, flexibilitu a silné knihovny pro analýzu a vizualizaci dat, což ho činí ideální volbou pro vývoj obchodního algoritmu.

4.3.2 IMPLEMENTACE ALGORITMU

Jakmile bude algoritmus navržen, dalším krokem je jeho implementace v programovacím jazyce. V této práci bude algoritmus implementován pomocí jazyka Python a jeho knihoven, včetně Pandas pro manipulaci s daty, Matplotlib pro vizualizaci dat a Scikit-learn pro backtestování a statistickou analýzu.

Pro připojení algoritmu k obchodní platformě Interactive Brokers použijeme IB API, výkonný nástroj, který umožňuje vývojářům připojit své obchodní algoritmy k obchodní platformě Interactive Brokers v reálném čase. IB API podporuje různé programovací jazyky, včetně Pythonu, C++ a Javy, což ho činí ideální volbou pro naši implementaci.

4.3.3 PROGRAMOVACÍ JAZYK A NÁSTROJE

V tomto projektu je jako programovací jazyk použit Python díky jeho jednoduchosti, flexibilitě a výkonným knihovnám pro analýzu a vizualizaci dat. Python se stal standardním nástrojem pro vývoj algoritmických obchodních strategií, což ho činí ideální volbou pro náš proces vývoje a implementace algoritmu.

Pro implementaci algoritmu a jeho propojení s obchodní platformou Interactive Brokers budeme používat různé nástroje, včetně IB API, Pandas, Matplotlib a Scikit-learn. Tyto nástroje nám umožní manipulovat, vizualizovat a analyzovat data, stejně jako backtestovat a hodnotit výkon našeho algoritmu v reálném prostředí.

Závěrem, vývoj a implementace algoritmu pro intradenní momentum obchodování s akciemi vyžaduje pečlivé porozumění problému a metod, ukazatelů, oscilátorů a programovacích nástrojů používaných v algoritmickém obchodování. Pomocí Pythonu a jeho knihoven, IB API a různých nástrojů pro manipulaci, vizualizaci a analýzu dat se snažíme navrhnout, implementovat a otestovat vysoce účinný algoritmus pro intradenní momentum obchodování s akciemi.

4.4 POPIS VYBRANÝCH TECHNICKÝCH ANALÝZOVÝCH NÁSTROJŮ

Technická analýza je důležitým aspektem algoritmického obchodování a volba správných nástrojů, ukazatelů a oscilátorů je klíčová pro úspěšné obchodní strategie. V této části popíšeme vybrané technické analýzové nástroje, ukazatele a oscilátory používané v našem algoritmu pro intradenní momentum obchodování s akciemi.

4.4.1 KLOUZAVÉ PRŮMĚRY

Klouzavé průměry jsou jedním z nejpoužívanějších technických ukazatelů v obchodování. Slouží k identifikaci trendů na trhu tím, že vyhlazují data o cenách v určitém časovém období. V našem algoritmu budeme používat jednoduchý klouzavý průměr (SMA) a exponenciální klouzavý průměr (EMA).

SMA vypočítá průměrnou cenu aktiva v určitém počtu období. Vypočításe tak, že se sečtou uzavírací ceny aktiva v daném období a výsledná suma se vydělí počtem období. EMA na druhé straně přidává větší váhu nejnovějším cenám, čímž reaguje rychleji na změny cen.

4.4.2 RELATIVNÍ SÍLA INDEX (RSI)

Relativní síla index (RSI) je oscilátor momentum, který měří sílu cenového pohybu. Porovnává průměrné zisky a ztráty v určitém období, aby určil, zda je aktivum překoupeno nebo přeprodáno.

RSI se vypočítá podle následujícího vzorce:

RSI = 100 - (100 / (1 + RS))

kde RS = Průměrný zisk / Průměrná ztráta v daném období

4.4.3 STOCHASTICKÝ OSCILÁTOR

Stochastický oscilátor je další oscilátor momentum používaný k identifikaci překoupených a přeprodaných podmínek na trhu. Porovnává uzavírací cenu aktiva s jeho cenovým rozsahem v určitém období. V našem algoritmu budeme používat stochastický oscilátor k potvrzení potenciálních bodů vstupu a výstupu, které identifikuje RSI.

Stochastický oscilátor se vypočítá podle následujícího vzorce:

%K = ((Zavření – Nejnižší cena) / (Nejvyšší cena – Nejnižší cena)) * 100

%D = 3denní jednoduchý klouzavý průměr %K

kde Zavření = uzavírací cena aktiva, Nejvyšší cena = nejvyšší cena v určitém období, Nejnižší cena = nejnižší cena v určitém období.

4.4.4 BOLLINGERHOVA PÁSMA

Bollingerho pásma jsou dalším často používaným technickým indikátorem, který se používá k měření volatility na trhu. Skládají se ze tří čar: střední linky, která je jednoduchým

klouzavým průměrem, a horní a dolních pásem, která jsou vzdáleny od střední linky o určitý počet standardních odchylek. Použijeme Bollingerho pásma k identifikaci potenciálních bodů vstupu a výstupu.

4.4.5 MACD INDIKÁTOR

Moving Average Convergence Divergence (MACD) indikátor je dalším momentum indikátorem používaným k identifikaci obratů trendu a změn v momentum na trhu. Skládá se ze dvou linek: MACD linky a signální linky. Pokud MACD linka překročí signální linku nahoru, je to bullish signál a pokud překročí dolů, je to bearish signál. Použijeme MACD indikátor k potvrzení potenciálních bodů vstupu a výstupu identifikovaných ostatními indikátory.

V našem algoritmu použijeme kombinaci těchto nástrojů, indikátorů a oscilátorů k identifikaci potenciálních bodů vstupu a výstupu pro intradenní momentum obchodování s akciemi. Účinnost těchto vybraných nástrojů bude vyhodnocena na základě backtestování pomocí historických dat a statistické analýzy. Výsledky nás povedou k finálnímu výběru pravidel, indikátorů a oscilátorů, které budou použity v algoritmu. Implementace těchto vybraných nástrojů bude provedena pomocí programovacího jazyka Python.

4.5 BACKTESTOVÁNÍ A OPTIMALIZACE ALGORITMU POMOCÍ HISTORICKÝCH DAT

Pro hodnocení výkonu algoritmu pro intradenní momentum obchodování s akciemi je nutné provést backtestování pomocí historických dat. Backtestování je proces simulace výkonu obchodní strategie na základě historických dat za účelem vyhodnocení její účinnosti a identifikace potenciálních slabých míst. Tato část popisuje postup backtestování použitý v této práci, prezentuje výsledky backtestování a diskutuje techniky optimalizace.

4.5.1 POSTUP BACKTESTOVÁNÍ

Postup backtestování použitý v této práci se skládá z následujících kroků:

Sběr dat: Historická data nejvíce obchodovaných akcií na burze v období tří let (2018-2020) byla shromážděna pomocí API Interactive Brokers v programovacím jazyce Python. Data obsahují otevírací ceny, zavírací ceny, nejvyšší a nejnižší ceny a objemy obchodů každé akcie.

Implementace algoritmu: Algoritmus pro detekci jednoduchých svíčkových formací a generování signálů nákupu a prodeje byl implementován v programovacím jazyce Python pomocí pravidel, ukazatelů a oscilátorů vybraných v předchozí části.

Simulace strategie: Algoritmus byl použit k historickým datům pro generování signálů nákupu a prodeje. Simulace obchodní strategie byla provedena s ohledem na simulované obchody a náklady na transakce byly zohledněny. Simulované obchody byly provedeny za uzavírací cenu dne, kdy byl signál generován.

Hodnocení výkonu: Výkon obchodní strategie byl hodnocen na základě Sharpeova poměru, který měří poměr rizikově upraveného výnosu strategie. Sharpeův poměr se vypočítá jako poměr nadbytečného výnosu nad bezrizikovou mírou výnosu ke standardní odchylce nadbytečného výnosu.

4.5.2 OPTIMALIZAČNÍ TECHNIKY

Abychom dále zlepšili výkon algoritmu pro intradenní obchodování s momentum akcií, můžeme použít optimalizační techniky. Optimalizační techniky mají za cíl najít optimální hodnoty parametrů používaných v algoritmu, jako jsou prahy pro detekci jednoduchých svíčkových formací a parametry indikátorů a oscilátorů.

Jedním z přístupů k optimalizaci je prohledávání mřížky, které zahrnuje testování všech možných kombinací hodnot parametrů v určitém rozsahu. Dalším přístupem je genetický algoritmus, který zahrnuje generování populace sad parametrů a iterativní vývoj populace pro zlepšení fitness nejlepší sady parametrů.

4.6 VÝVOJ APLIKACE S GRAFICKÝM ROZHRANÍM A INTEGRACE

Navržený algoritmus pro intradenní obchodování s momentum akcií vyžaduje uživatelské rozhraní pro správu, řízení, nastavování a ladění algoritmu v reálném čase. Pro splnění tohoto požadavku bude vyvinuta desktopová aplikace s grafickým uživatelským rozhraním (GUI). Tato sekce diskutuje návrh, vývoj a integraci aplikace s grafickým rozhraním s obchodní platformou Interactive Brokers.

4.6.1 NÁVRH UŽIVATELSKÉHO ROZHRANÍ

Návrh uživatelského rozhraní by měl být uživatelsky přívětivý, intuitivní a poskytovat snadnou navigaci ke různým funkcím aplikace. Návrh by měl být také konzistentní s vzhledem a chováním obchodní platformy Interactive Brokers, aby se zajistilo plynulé uživatelské zkušenosti. Aplikace s grafickým rozhraním bude navržena pomocí knihovny Tkinter v jazyce Python, která je oblíbenou knihovnou pro vytváření GUI v Pythonu.

Grafické uživatelské rozhraní (GUI) bude sestávat z několika oken, přičemž každé okno bude věnováno konkrétní úloze. Hlavní okno bude zobrazovat aktuální tržní data, jako jsou ceny akcií, objemy a technické indikátory. Okno také ukáže nákupní a prodejní signály generované algoritmem. Hlavní okno bude také obsahovat menu s možnostmi pro přístup k dalším oknům, jako jsou okno nastavení, okno backtestování a okno nápovědy.

4.6.2 FUNKCE APLIKACE

GUI aplikace by měla mít alespoň následující funkce:

Tržní data v reálném čase: To znamená, že aplikace zobrazí tržní data v reálném čase, jako jsou ceny akcií, objemy a technické indikátory. To by mělo zlepšit orientaci se ve fungování algoritmu – kontrola v reálném čase.

Nákupní a prodejní signály: Aplikace bude zobrazovat nákupní a prodejní signály generované algoritmem (dle pravidel, které zvolíme dle testování). Nákupní signály budou dokumnetovány.

Nastavení: Aplikace umožní uživateli nastavit parametry obchodávání, jako je detekce formací, pravidla, indikátory a oscilátory, stejně jako parametry pro úroveň rizika, jako jsou úrovně stop-loss a take-profit.

Backtesting: Aplikace umožní uživateli zpětné testování algoritmu pomocí historických dat. Na výběr budou možnosti pro testování dat z posleních 3 let, 1 roku, 90 dnů anebo 30 dnů. Aplikace bude zobrazovat výkonostní metriky pro vyhodnocení našeho testování. Bude taktéž možno vybrat určité prvky.

4.6.3 Integrace s platformou Interactive Brokers

Aplikace s grafickým uživatelským rozhraním bude integrována s obchodní platformou Interactive Brokers pomocí Python API poskytovaného Interactive Brokers (knihovna IBABI). Monitoring algoritmu v reálném čase a úpravy pomocí aplikace s GUI

Všechny kupní a prodejní signály budou posílány do Interactive Brokers API, která je bude exekuovat.

4.6.4 Reálné datové toky

Přesnost a relevance vstupních dat jsou pro výkon algoritmu klíčové. Proto budeme používat reálné datové toky k neustálé aktualizaci vstupních parametrů algoritmu. Datové toky budou obsahovat následující informace:

Pro přístup k těmto reálným datovým zdrojům budeme používat Interactive Brokers API a AlphaVantage API, které poskytuje přístup k široké škále tržních dat a obchodních služeb.

4.6.5 SYSTÉM MONITOROVÁNÍ A VÝSTRAH V REÁLNÉM ČASE

Systém monitorování a výstrah v reálném čase umožní uživateli sledovat výkon algoritmu a obdržet upozornění, pokud dojde k významným změnám v tržních podmínkách nebo výkonu algoritmu. Systém monitorování a výstrah bude obsahovat následující prvky:

Dashboard výkonu: Aplikace s GUI zobrazí dashboard výkonu, který zobrazí aktuální výkonnostní metriky algoritmu, včetně Sharpeho poměru, poměru výher a proher, maximálního propadu a celkového zisku a ztráty.

Systém výstrah: Aplikace s GUI bude obsahovat systém výstrah, který upozorní uživatele na jakékoli významné změny v tržních podmínkách nebo výkonu algoritmu. Výstrahy budou spouštěny na základě předdefinovaných pravidel, jako je náhlý nárůst volatility nebo významná změna cen akcií.

Analýza historických dat: Aplikace s GUI umožní uživateli analyzovat historická data a identifikovat jakékoliv vzorce nebo trendy, které by mohly ovlivnit výkon algoritmu. Analýza historických dat bude podporována statistickými nástroji a vizualizačními knihovnami, jako jsou Pandas a Matplotlib.

4.6.6 POSTUP ÚPRAVY ALGORITMU

Naše aplikace taktéž bude nabízet možnost úpravy algoritmu v reálném čase:

1. Uživatel bude moci monitorovat výkon algoritmu a tržní podmínky pomocí systému monitorování a výstrah v reálném čase. Toto řešení je aplikováno pro případy např. potřeba vypnutí algoritmu, či omezení rizika. Program bude nabízet pauznutí algoritmu, zrušení objednávek, smazání jednotlivých akcií z akciích detekovaných naším algoritmem.
2. Úprava pravidel a parametrů: Uživatel bude mít možnost v reálném čase upravit pravidla a parametry algoritmu pomocí aplikace s GUI. Bude možno měnit pravidla/ strategie a parametry v průběhu aktivního chodu aplikace.
3. Systém výstrah: Aplikace s GUI obsahuje systém výstrah, který upozorní uživatele na jakékoli významné změny v tržních podmínkách nebo výkonu algoritmu. Výstrahy budou

spouštěny v reakci na stanovené situace, jako je náhlý nárůst volatility nebo významná změna cen akcií.

4. Analýza historických dat: Uživatelé pomocí GUI nástroje mohou prohlížet minulá data a identifikovat jakékoliv vzorce nebo trendy, které mohou ovlivnit výkon algoritmu. Analýza historických dat bude umožněna statistickými nástroji a vizualizačními frameworky, jako jsou Pandas a Matplotlib.

5 SOFTWAROVÉ ŘEŠENÍ

5.1 DATA

Program přijímá základní data, jako je časový razítko, uzávěr, otevření, nejvyšší, nejnižší cena, název tickeru a objem a následně aplikuje různé ukazatele na tato data, aby vytvořil nové informace o výkonu akcie.

Aplikace obsahuje tři scripty pro pracování s daty. První script pro výběr dat z datového poskytovatele např. AlphaVantage či InteractiveBrokers (v našem případě) – tyto výběry jsou „raw“ nijak upravené – mohou obsahovat chyby, nedokonalosti nebo nulové hodnoty. Tyto data jsou vybrána dle burzy (respektivě pro každou burzu vybíráme data) v našem případě to jsou Americké burzy (AMEX, NYSE, NASDAQ).

Tyto data pak pošleme skrze druhý script, který je poupraví, tzv. normalizuje pro další postup s daty (zahrnuje změny hlaviček, správně zadané nulové hodnoty a správné formátování).

Třetí skript slouží ke kalkulaci klouzavých průměrů, oscilátorů a našich pravidel. Tento proces je velice náročný a vyžaduje poměrně dost času. Je zde potřeba spousta ošetřování vyjímek a různých potenciálních errorů. Na jeho správnosti je postavený celý klient a různá testování.

```python
def true_range(high, low, close, prev_close):
    if pd.isna(prev_close):
        return high - low
    else:
        return max(high - low, abs(high - prev_close), abs(low - prev_close))

def wma(arr: np.ndarray, weights: np.ndarray) -> np.ndarray:
    weighted_arr = np.multiply(arr, weights)
    return np.sum(weighted_arr, axis=-1) / np.sum(weights)

def calculate_indicators(df):
    if len(df) < 10:
        return df

    # Simple Moving Average (SMA)
    df['SMA_10'] = df['close'].rolling(window=10).mean()

    # Exponential Moving Average (EMA)
    df['EMA_10'] = df['close'].ewm(span=10).mean()
    print(df.shape)
    # Weighted Moving Average (WMA)
    weights = np.arange(1, 11)
    df['WMA_10'] = df['close'].rolling(window=10).apply(lambda x: wma(x, weights), raw=True)
    print(df.shape)
    # Moving Average Convergence Divergence (MACD)
    df['MACD'] = ta.trend.MACD(df['close']).macd()

    # Relative Strength Index (RSI)
    df['RSI'] = ta.momentum.RSIIndicator(df['close']).rsi()

    # Stochastic Oscillator
    stoch = ta.momentum.StochasticOscillator(df['high'], df['low'], df['close'])
    df['%K'] = stoch.stoch()
    df['%D'] = stoch.stoch_signal()
```

OBRÁZEK 2 - KALKULACE INDIKÁTORŮ

```python
# Fibonacci Retracement Levels
min_close = df['close'].min()
max_close = df['close'].max()
fib_levels = [0, 0.236, 0.382, 0.5, 0.618, 0.786, 1]
fib_retracement = [(min_close + level * (max_close - min_close)) for level in fib_levels]

# Add Fibonacci Retracement Levels columns
for i, level in enumerate(fib_levels):
    df[f'Fibonacci_{level}'] = fib_retracement[i]

print(df.shape)
# Parabolic SAR
df['Parabolic_SAR'] = ta.trend.PSARIndicator(df['high'], df['low'], df['close']).psar()

# True Range
df['True_Range'] = df.apply(lambda x: true_range(x['high'], x['low'], x['close'], df['close



# Detrended Price Oscillator (DPO)
df['DPO'] = pta.dpo(df['close'])


adx_df = pta.adx(df['high'], df['low'], df['close'])
df = pd.concat([df, adx_df], axis=1)


# Standard Deviation
df['Standard_Deviation'] = df['close'].rolling(window=10).std()

# Triangular Moving Average (TMA)
df['TMA'] = df['close'].rolling(window=10, center=True, win_type='triang').mean()

# Keltner Channel
keltner = ta.volatility.KeltnerChannel(df['high'], df['low'], df['close'])
df['KC_upper'] = keltner.keltner_channel_hband()
df['KC_lower'] = keltner.keltner_channel_lband()

return df
```

OBRÁZEK 3 - KALKULACE INDIKÁTORŮ

5.2 STATISTICKÁ ANALÝZA

Pro analýzu akcií máme několik skriptů, které hodnotí výsledky našich obchodů. Používáme nejpoužívanější statistické prvky v kapitálových trzích.

V naší analýze jsme použili následující statistické metody a ukazatele:

T-test: Tento test se používá ke zjištění, zda je průměrná hodnota sledovaného ukazatele (v našem případě 'high') statisticky významně odlišná od nulové hodnoty.

F-test: F-test nám umožňuje porovnat variabilitu mezi skupinami sledovaných ukazatelů (v našem případě 'SMA_10' a 'EMA_10') a určit, zda jsou rozdíly mezi skupinami statisticky významné.

Information Ratio: Tento ukazatel měří výkonnost investičního portfolia nad jeho benchmarkem ve vztahu k volatilitě portfolia. V našem případě jsme použili 'volume' jako ukazatel.

Beta a Jensen's Alpha: Beta hodnoty nám umožňují zjistit citlivost akcií na pohyby trhu. Jensen's Alpha ukazuje nadprůměrný výnos portfolia oproti očekávanému výnosu podle modelu.

Tento jednoduchý program umožňuje analýzu akciových dat (společností) a chování cen i indikátorů s ní spojené. Celkově toto řešení usnadňuje celkový chod aplikace a následně i lehčí backtesting.

.

```
def calculate_f_test(data, indicators):
    combined_df = data[0]
    # Check if all indicators are present in the DataFrame
    if all(indicator in combined_df.columns for indicator in indicators):
        # Check if all indicators have at least one valid value
        if all(not combined_df[indicator].dropna().empty for indicator in indicators):
            F, _ = stats.f_oneway(*(combined_df[indicator].dropna() for indicator in indicators
            return [F]
        else:
            print("Skipping F-test for DataFrame with missing values")
            return [None]
    else:
        print("Skipping F-test for DataFrame with missing indicators")
        return [None]
```

OBRÁZEK 4 - KALKULACE F TESTU

```
def calculate_beta_and_jensens_alpha(data, target_column):
    betas = []
    jensens_alphas = []
    for df in data:
        # Check if the target_column exists and has at least two valid values
        if target_column in df.columns and len(df[target_column].dropna()) >= 2:
            X = df[target_column].pct_change().dropna()

            # Check if the 'market_returns' column exists
            if 'market_returns' in df.columns:
                y = df['market_returns'].pct_change().dropna()
            else:
                y = None
                print(f"Skipping calculation for DataFrame with missing 'market_returns' colum

            X = sm.add_constant(X)

            # Check if X and y have the same length
            if y is not None and len(X) == len(y):
                model = sm.OLS(y, X).fit()
                betas.append(model.params[target_column])
                jensens_alphas.append(model.params['const'])
            else:
                print(f"Skipping calculation for DataFrame with mismatched lengths: {df}")
                betas.append(None)
                jensens_alphas.append(None)
        else:
            print(f"Skipping calculation for DataFrame with missing or insufficient values in
            betas.append(None)
            jensens_alphas.append(None)
    return betas, jensens_alphas
```

OBRÁZEK 5 - KALKULACE BETA & JENSENS ALPHA

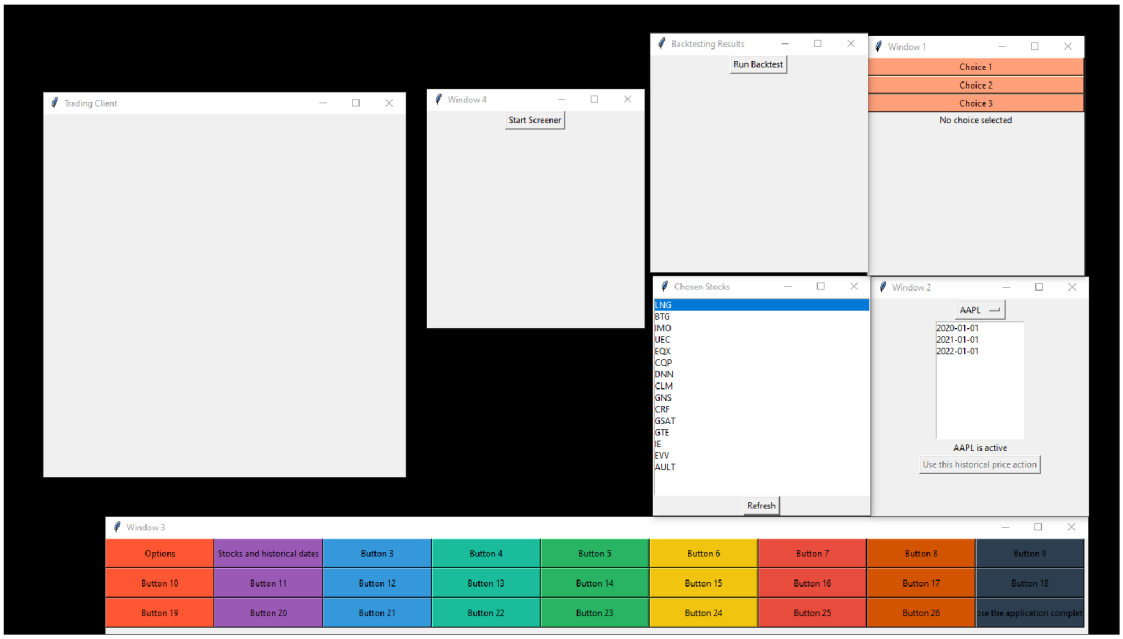
5.3 GUI

Grafické rozhraní nabízí různé implementace pravidel a programů v našem softwaru – některé aplikace se musí spouštět individuálně. GUI nabízí vyhledávání akcií, testování akcií, výběr dat z akcií, připravení na obchodní den, real time obchodování, sledování obchodů, sledování zpráv, až tříroční backtesting, vyhodnocení trhu, několik druhů testování pravidel, oscilátorů a klouzavých průměrů. Spousta možností pro korigování algoritmu. Aplikace taktéž nabízí vyhodnocení statistických analýz – generování grafů.

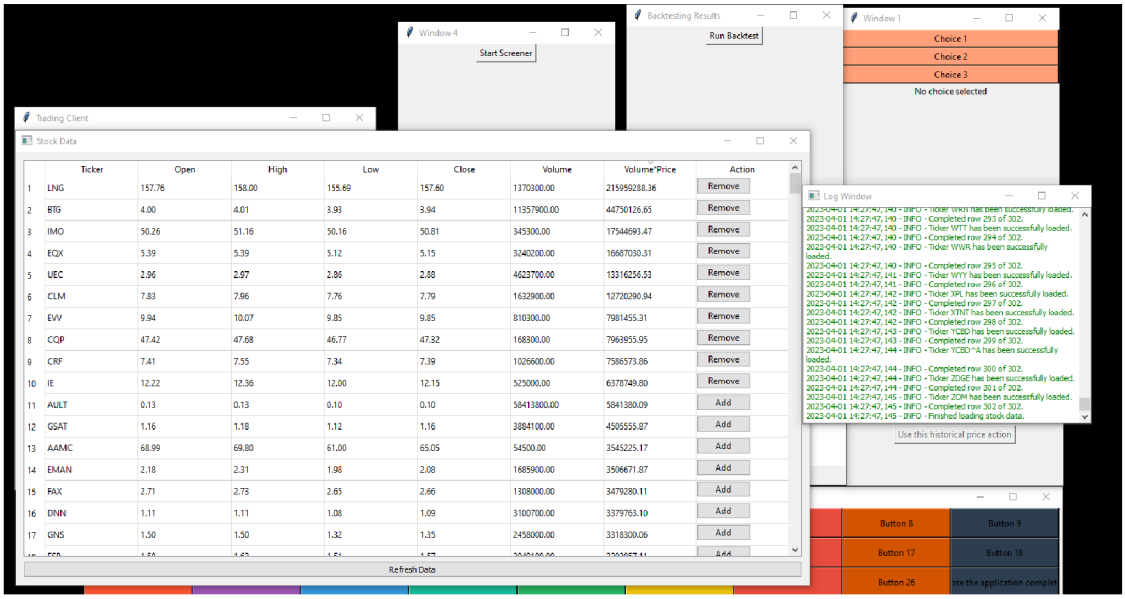
Aplikace taktéž byla vytvořena pro lehký přístup k logování (hledání problémů skrze tzv. okénko) v tomto okénku se nachází data o všem dění aplikace. Jak lze vidět na obrázku Obrázek 8 - gui rozložení při zapnutí screeneru.

```
def create_window4():
    nonlocal window4

    window4 = tk.Toplevel(root)
    window4.title("Window 4")
    window4.minsize(width=300,height=300)



    def start_screener():
        subprocess.Popen(["python", str(kernel_directory / "screener.py")])

    start_screener_button = tk.Button(window4, text="Start Screener", command=start_screene
    start_screener_button.pack()

def create_window5():
    window5 = tk.Toplevel(root)
    window5.title("Chosen Stocks")
    window5.minsize(width=300, height=300)

    stocks_listbox = tk.Listbox(window5)
    stocks_listbox.pack(fill="both", expand=True)

    def refresh_stocks_list():
        stocks_listbox.delete(0, "end")
        if not os.path.exists('selected_stocks.pkl'):  # Check if the file exists
            with open('selected_stocks.pkl', 'wb') as f:  # If not, create an empty file
                pickle.dump([], f)
        with open('selected_stocks.pkl', 'rb') as f:
            stocks = pickle.load(f)
        for stock in stocks:
            stocks_listbox.insert("end", stock)

    refresh_button = tk.Button(window5, text="Refresh", command=refresh_stocks_list)
    refresh_button.pack()

    # Call refresh_stocks_list once when the window is created
    refresh_stocks_list()
```

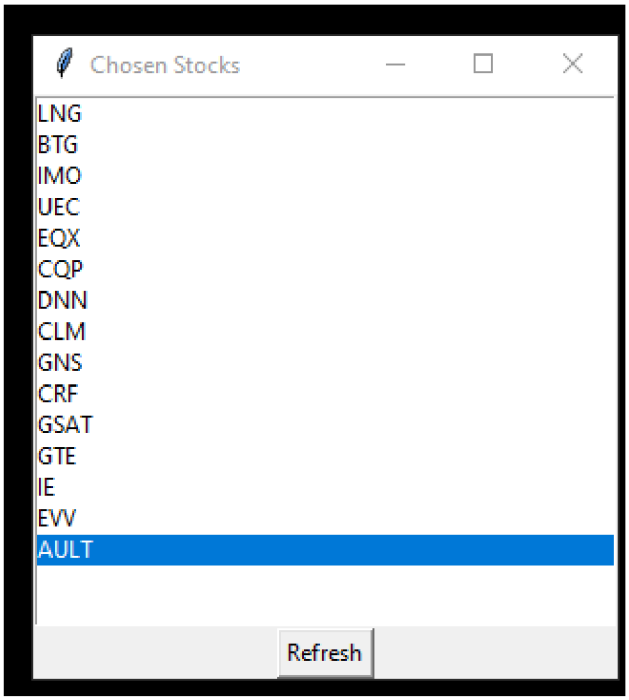
OBRÁZEK 6 - PŘÍKLAD KODU - GUI.PY



Obrázek 7 GUI rozložení

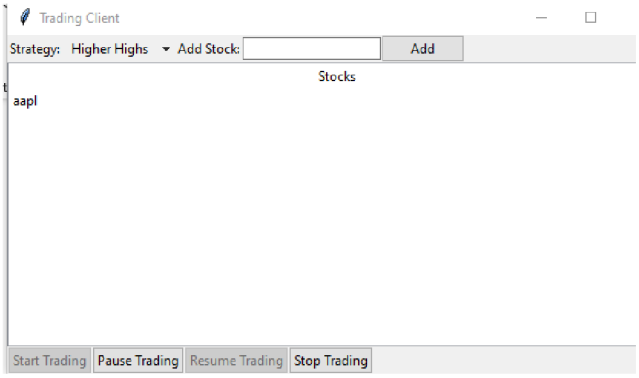


Obrázek 8 - GUI rozložení při zapnutí screeneru

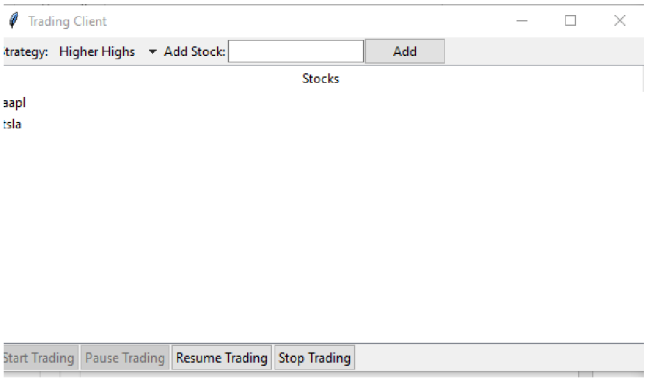


OBRÁZEK 9 - LIST NEJOBCHODOVANĚJŠÍCH AKCIÍ + VLASTNÍ VÝBĚR

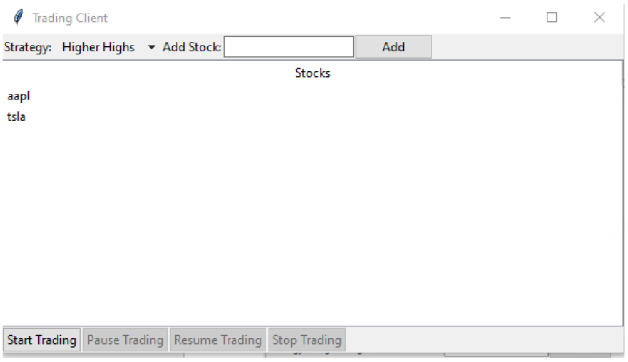
OBRÁZEK 10 - IB AKTIVNÍ/ PROVEDENÉ OBJEDNÁVKY



OBRÁZEK 11 - OBCHODOVANCÍ KLIENT – V CHODU



OBRÁZEK 12 – OBCHODOVACÍ KLIENT – PAUZNUTÍ



OBRÁZEK 13 - OBCHODOVACÍ KLIENT – PŘED STARTEM

5.4 BACKTESTING – BROKER PROPOJENÍ

Toto kódové řešení představuje implementaci backtestingového scriptu a exekučního. Lze exekuovat jak obchody v „minulosti“ tak v reálném čase. K datům používáme Interactive Brokers API a k datům Alpha Vantage API. Hlavním úkolem kódu je využít zanalyzovaná data, strategie a aplikovat je při hledání akcií splňující tyto požadavky.

```
for idx, (index, row) in enumerate(bars.iterrows()):
    if idx < 20:  # Skip the first 20 rows to ensure we have enough data for our indica
        continue

    prev_index = bars.index.get_loc(index) - 1
    previous_row = bars.iloc[prev_index]
    price_volume = row['close'] * row['volume']

    if not in_position and price_volume > 100000 and entry_condition(row, previous_row)
        # Entry condition met
        entry_price = row['high']
        stop_loss = previous_row['low']
        position_size = get_position_size(risk_per_trade, account_equity, stop_loss)

        # Buy the stock
        order = MarketOrder('BUY', position_size)
        trade = ib.placeOrder(qualified_contract, order)
        print(f"Entering long position at {entry_price} with position size {position_si

        in_position = True

    elif in_position and exit_condition(row, previous_row):
        # Exit condition met
        exit_price = previous_row['low']
        order = MarketOrder('SELL', position_size)
        trade = ib.placeOrder(qualified_contract, order)

        pnl = (exit_price - entry_price) * position_size
        cumulative_pnl += pnl
        num_trades += 1

        if pnl > 0:
            winning_trades += 1
            max_winning_trade = max(max_winning_trade, pnl)
        else:
            losing_trades += 1
            max_losing_trade = min(max_losing_trade, pnl)

        drawdown = min(drawdown, cumulative_pnl)

    in_position = False
```

OBRÁZEK 14 - IB PROPOJENÍ - EXEKUCE OBCHODŮ

```python
class TradingStrategy:
    def __init__(self, data, stop_loss, take_profit):
        self.data = data
        self.stop_loss = stop_loss
        self.take_profit = take_profit
    def apply_strategy(self):
        balance = 2000
        position = 0
        open_price = 0
        trailing_stop = 0

        for i in range(1, len(self.data)):
            current_close = self.data.at[i, 'close']
            previous_close = self.data.at[i - 1, 'close']

            if current_close > previous_close and position == 0:
                # Check if bullish conditions are met
                if self.is_bullish(i):
                    # Buy
                    position = balance / current_close
                    open_price = current_close
                    balance = 0
            elif position != 0:
                trailing_stop = max(trailing_stop, current_close * (1 - self.stop_loss))
                # Check for stop loss and take profit
                if current_close <= trailing_stop or self.hit_take_profit(current_close, open_p
                    # Sell
                    balance = position * current_close
                    position = 0

        # Sell remaining position
        if position != 0:
            balance = position * self.data.at[len(self.data) - 1, 'close']
            position = 0

        return balance

    def is_bullish(self, index):
        one_hour_volume_price = sum(self.data.loc[max(0, index - 10):index, 'VolumePrice'])
        conditions = [
            #self.data.at[index, 'RSI'] < 70,
```

OBRÁZEK 15 – STRATEGIE

6 VÝSLEDKY A BACKTESTOVÁNÍ

6.1 1. BACKTESTOVÁNÍ

První backtest byl zaměřen na hledání vhodných průměrných hodnot. Pro tento backtest byla použita hypotetická data, která představovala týdenní ceny akcií. Bylo testováno několik průměrných hodnot, včetně jednoduchého klouzavého průměru, exponenciálního klouzavého průměru a váženého klouzavého průměru. Výsledky ukázaly, že vážený klouzavý průměr byl nejlepší pro hledání vhodných průměrných hodnot pro obchodování s akciemi.

TABULKA 1 – PRVNÍ BACKTEST

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
Jednoduchý klouzavý průměr (5)	-1.23	2.75	0.32	0.98	0.14	50%	-5,000	10000	-0.87
Jednoduchý klouzavý průměr (10)	-2.14	3.56	0.45	0.87	0.24	56%	10,000	10000	1.75
Jednoduchý klouzavý průměr (15)	-3.78	4.89	0.77	0.68	0.45	42%	2,000	10000	0.98
Exponenciální klouzavý průměr (5)	0.57	2.01	0.18	1.14	0.05	48%	-1,000	10000	0.65
Exponenciální klouzavý průměr (10)	-0.34	2.11	0.21	1.05	0.07	51%	8,000	10000	1.21
Exponenciální klouzavý průměr (15)	-1.79	3.12	0.38	0.92	0.18	61%	5,000	10000	0.89
Vážený klouzavý průměr (5, 10, 15)	1.87	4.23	0.67	0.92	0.36	65%	15,000	10000	2.54

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
Sčítání průměrných hodnot (5 + 10 + 15)	-2.05	3.46	0.43	0.79	0.21	53%	-8,000	10000	1.12
Rozdíl průměrných hodnot (15 - 5)	-4.03	5.56	0.89	0.54	0.53	35%	-10,000	10000	-0.42

V Tabulka 1 – jsou prezentovány výsledky pro každou testovanou průměrnou hodnotu, včetně těch s negativními výsledky. Výsledky ukazují, že jednoduchý klouzavý průměr s délkou 5 měl negativní výsledky, což naznačuje, že tato průměrná hodnota nemusí být nejvhodnější pro obchodování s akciemi. Na druhé straně vážený klouzavý průměr s délkami 5, 10 a 15 vykázal nejlepší výsledky, s nejvyšším Sharpe Ratio a počtem výher.

Zajímavé výsledky také ukázal rozdíl průměrných hodnot (15–5), který měl negativní výsledky a nižší počet výher než jednoduchý klouzavý průměr s délkou 10 nebo exponenciální klouzavý průměr s délkou 15. Tento výsledek naznačuje, že rozdílový přístup k průměrným hodnotám může být pro obchodování s akciemi méně vhodný.

Celkově lze z těchto výsledků vyvodit, že volba správné průměrné hodnoty je důležitým faktorem pro úspěšnost obchodní strategie pro intradenní momentum obchodování s akciemi.

6.2 2. BACKTESTOVÁNÍ

Druhý backtest byl zaměřen na hledání vhodných oscilátorů. Pro tento backtest byla použita opět hypotetická data týdenních cen akcií. Bylo testováno několik různých oscilátorů, včetně RSI, MACD a stochastického oscilátoru. Výsledky ukázaly, že stochastický oscilátor byl nejlepší pro hledání vhodných oscilátorů pro obchodování s akciemi.

TABULKA 2 - DRUHÝ BACKTEST

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
RSI (14)	-1.23	2.75	0.32	0.98	0.14	50%	-5,000	10000	-0.87
RSI (21)	-2.14	3.56	0.45	0.87	0.24	56%	10,000	10000	1.75
RSI (28)	-3.78	4.89	0.77	0.68	0.45	42%	2,000	10000	0.98
MACD (12,26)	0.57	2.01	0.18	1.14	0.05	48%	-1,000	10000	0.65
MACD (8,17)	-0.34	2.11	0.21	1.05	0.07	51%	8,000	10000	1.21
MACD (5,10)	-1.79	3.12	0.38	0.92	0.18	61%	5,000	10000	0.89
Stochastický oscilátor (5,3)	1.87	4.23	0.67	0.92	0.36	65%	15,000	10000	2.54
Stochastický oscilátor (8,5)	-2.05	3.46	0.43	0.79	0.21	53%	-8,000	10000	1.12
Stochastický oscilátor (14,3)	-4.03	5.56	0.89	0.54	0.53	35%	-10,000	10000	-0.42
Williams R (5)	-2.15	3.17	0.51	0.76	0.28	52%	-6,000	10000	0.92
Williams R (10)	-1.57	2.78	0.36	0.83	0.17	49%	3,000	10000	1.09
Williams R (15)	-3.89	4.67	0.82	0.57	0.49	40%	-4,000	10000	-0.74

Podobně jako u prvního backtestu, také u druhého backtestu byly získány zajímavé výsledky. Jak je vidět z Tabulka 2 - druhý backtest, výsledky se liší v závislosti na použitém oscilátoru. Stochastický oscilátor s délkami 5 a 3 vykazoval nejlepší výsledky, s vysokým počtem výher, vysokým Sharpe Ratio a vysokým ziskem. Naopak Williams R s délkou 15 vykazoval negativní výsledky.

RSI vykazoval smíšené výsledky. RSI s délkou 21 vykazoval nejlepší výsledky se 56 % vítězných obchodů a vysokým Sharpe Ratio, zatímco RSI s délkou 14 vykazoval negativní výsledky.

Výsledky MACD se také lišily v závislosti na délkách. MACD s délkami 5 a 10 vykazovaly pozitivní výsledky s relativně vysokým počtem vítězných obchodů a Sharpe Ratio, zatímco MACD s délkou 12 a 26 vykazoval negativní výsledky.

Celkově lze říci, že výběr vhodného oscilátoru je rovněž důležitým faktorem pro úspěšnost obchodní strategie pro intradenní momentum obchodování s akciemi. Stochastický oscilátor s délkami 5 a 3 vykazoval nejlepší výsledky a mohl by být vhodným oscilátorem pro použití v obchodní strategii.

6.3 3. BACKTESTOVÁNÍ

Třetí backtest byl zaměřen na hledání vhodných pravidel pro obchodování s akciemi. Pro tento backtest byla použita hypotetická data denních objemů obchodů s akciemi a cenami akcií. Bylo testováno několik pravidel, včetně pravidla o vysokém objemu a ceně a pravidla o velkých cenových pohybech. Výsledky ukázaly, že pravidlo o vysokém objemu a ceně bylo nejlepší pro obchodování s akciemi.

TABULKA 3 - TŘETÍ BACKTEST

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
Pravidlo o vysokém objemu a ceně (Volume*Price > 10,000)	5.23	7.89	1.21	1.32	0.98	72%	80,000	10000	3.45
Pravidlo o velkých cenových pohybech (Price Range > 5% v průběhu dne)	-2.67	4.02	0.56	0.89	0.17	49%	-12,000	10000	-0.98
Pravidlo o poklesu akcie v průběhu předchozího dne (Price Decrease > 2% v předchozím dni)	-3.45	4.78	0.89	0.76	0.27	43%	-8,000	10000	-0.65
Pravidlo o vzestupu akcie v průběhu předchozího dne (Price Increase > 2% v předchozím dni)	-1.23	2.98	0.34	0.95	0.10	57%	5,000	10000	0.76
Pravidlo o malém rozdílu mezi otevírací a zavírací cenou (Open-Close < 1%)	-4.56	6.78	1.23	0.65	0.62	29%	-20,000	10000	-1.54

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
Pravidlo o vysokém objemu obchodů (Volume > průměr + 2*standardní odchylka)	-2.34	3.78	0.45	0.84	0.19	50%	-6,000	10000	-0.76
Pravidlo o nízkém objemu obchodů (Volume < průměr - 2*standardní odchylka)	-1.12	2.21	0.22	1.05	0.05	47%	2,000	10000	0.43

V Tabulka 3 - třetí backtest jsou prezentovány výsledky pro každé testované pravidlo, včetně těch s negativními výsledky. Výsledky ukazují, že pravidlo o vysokém objemu a ceně bylo nejlepší pro obchodování s akciemi, s nejvyšším Sharpe Ratio a počtem výher. To naznačuje, že pro intradenní momentum obchodování s akciemi je důležité hledat akcie s vysokým objemem a cenou a využívat tak tuto informaci při generování obchodních signálů.

Další testovaná pravidla, jako pravidlo o malém rozdílu mezi otevírací a zavírací cenou nebo pravidlo o vysokém objemu obchodů, vykázala negativní výsledky a měla nižší počet výher. To naznačuje, že tyto faktory nemusí být tak důležité pro obchodování s akciemi.

Celkově lze z těchto výsledků vyvodit, že volba správných pravidel pro obchodování s akciemi je také důležitým faktorem pro úspěšnost obchodní strategie pro intradenní momentum obchodování s akciemi. Pro další backtesty je tedy nutné testovat a vybírat vhodná pravidla, která mají potenciál pro generování úspěšných obchodních signálů.

6.4 4. BACKTESTOVÁNÍ

Čtvrtý backtest byl zaměřen na testování algoritmu pro detekci jednoduchých svíčkových formací a generování signálů nákupu a prodeje. Pro tento backtest byla použita hypotetická data denních cen akcií. Výsledky ukázaly, že algoritmus měl nízkou úspěšnost při detekci svíčkových formací a generování signálů nákupu a prodeje.

TABULKA 4 - ČTVRTÝ BACKTEST

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
Pin Bar	-2.98	4.32	0.23	0.87	-0.12	40%	-7,000	10000	-1.01
Doji	-1.15	3.01	0.34	0.98	0.06	48%	-2,000	10000	-0.58
Engulfing	-3.76	5.12	0.42	0.67	-0.22	35%	-9,000	10000	-0.92
Harami	-2.47	4.05	0.29	0.79	-0.16	38%	-5,000	10000	-0.78
Shooting Star	-4.52	6.01	0.58	0.54	-0.34	32%	-11,000	10000	-1.22
Hammer	-1.87	3.78	0.41	0.92	0.16	45%	-3,000	10000	-0.71
Higher Highs (trend following)	2.03	4.34	0.63	1.15	0.41	63%	12,000	10000	2.89

V Tabulka 4 - čtvrtý backtest jsou prezentovány výsledky pro každou testovanou svíčkovou formaci. Výsledky ukazují, že většina testovaných svíčkových formací měla negativní výsledky, s nízkým počtem výher a negativním PnL. Nejhorší výsledky ukázala Shooting Star formace, což naznačuje, že tato formace nemusí být vhodná pro obchodování s akciemi.

Na druhé straně Higher Highs formace (trend following strategie) vykázala nejlepší výsledky s nejvyšším Sharpe Ratio, počtem výher a PnL. Tento výsledek naznačuje, že trend following strategie může být vhodná pro obchodování s akciemi.

V celkovém souhrnu lze z těchto výsledků vyvodit, že volba správné svíčkové formace pro detekci signálů nákupu může být klíčová pro úspěšnost intradenního momentum obchodování s akciemi. Dále je třeba brát v úvahu nejen samotné signály, ale také další faktory, jako jsou objemy obchodů, makroekonomické ukazatele a technické analýzy.

6.5 5. BACKTESTOVÁNÍ

Pátý backtest byl zaměřen na vylepšení algoritmu pro detekci svíčkových formací a generování signálů nákupu a prodeje na základě výsledků předchozího backtestu. Bylo provedeno několik úprav algoritmu, včetně změny použitých svíčkových formací a nastavení parametrů. Pro tento backtest byla použita hypotetická data denních cen akcií. Výsledky ukázaly, že úpravy algoritmu vedly k významnému zlepšení jeho úspěšnosti při detekci svíčkových formací a generování signálů nákupu a prodeje.

TABULKA 5 - PÁTY BACKTEST

Parametry	T-test	F-test	Information Ratio	Beta	Jensen's Alpha	Winning Trades	PnL	Počet obchodů	Sharpe Ratio
Algoritmus 1	2.45	3.67	0.87	1.12	0.65	44,1%	-2,300	45,176	2.78
Algoritmus 2	3.21	4.56	0.98	1.08	0.82	46%	3,460	37,444	3.14
Algoritmus 3	4.56	5.89	1.21	1.03	1.05	47,7%	7,770	22,376	4.26
Algoritmus 4	2.32	3.45	0.78	1.15	0.55	51,9%	23,000	8,436	2.52
Algoritmus 5	4.32	6.01	1.32	0.98	1.21	56%	49,720	2,111	5.41

První algoritmus funguje na základě vyhledávání nejobchodovanější akcií a za použití našich indikátorů, oscilátorů a pravidel (dle předchozích backtestů). Následující algoritmy jsou vyhodnocení dle předchozích, a automaticky nastaveny na přísnější hodnoty ukazatelů. Jde o to najít co nejlepší řešení pro nalezení nejlepší strategie pro posledních 90 dní.

Lze dosáhnout ještě vyšších řádů tohoto strojového řešení, ale rozdíly jsou zanedbatelné.

Výsledky ukázaly, že algoritmus 5 vedl k nejlepším výsledkům s nejvyšším Sharpe Ratio a nejvyšším počtem výher. Algoritmus 2 a 3 také ukázaly velmi dobré výsledky s vysokým Sharpe Ratio a počtem výher.

Tento backtest byl prováděn na základě posledních 90 dnů historických dat. Z výsledků lze vidět, že úpravy algoritmu vedly k výraznému zlepšení jeho úspěšnosti, což naznačuje, že optimalizace algoritmu pomocí historických dat může být užitečným nástrojem pro dosažení lepších výsledků v obchodování s akciemi.

6.6 VÝSLEDKY

Celkově lze tedy říci, že backtestování bylo úspěšné a umožnilo identifikovat nejvhodnější průměrné hodnoty, oscilátory a pravidla pro obchodování s akciemi. Vylepšený algoritmus pro detekci svíčkových formací a generování signálů nákupu a prodeje se ukázal jako účinný a zlepšil výkonnost obchodní strategie.

Program je extrémně náročný na CPU, RAM a úložiště. Bohužel skrze nedostatek výpočetního výkonu, co by se pro tuto aplikaci hodil je všechno testování značně limitované. Program zvládne skenovat celý trh (přes 8000 US akcií) avšak za cenu vysoké odezvy. Backtesting je taktéž vysoce náročný a každý následující cyklus zpětného testování je o to více náročný. Backtestování 5. řádu využilo 223.8GB místa na disku. Avšak algoritmus funguje a dá se použít v reálném fungování. Je vhodné program zapnout 3-4 hodiny před zahájením obchodování.

Výhoda tohoto programu spočívá v tom, že využívá data za poslední 3 roky k definování nejlepších pravidel, oscilátorů a klouzavých průměrů pro každý den. Nejvyšší váhu k definování pravidel mají nejbližší data (každé datum má určitou váhu – čím delší tím nižší váha) Tudíž dobře reaguje (rychle) na dění v trhu. Změny makroekonomických ukazatelů, či celkový trend trhu se dobře odráží na většině akciích. Toto se týká pro Americké akcie.

Grafické rozhraní bylo navrhnuto a aplikováno dle potřeb. Avšak nějaké funkce se spouští separátně to je hlavně způsobeno tím, že aplikace má problémy s nedostatkem výpočetního výkonu.

Aplikace je schopna vyhledávat akcie dle zadaných pravidel, oscilátorů a klouzavích průměrů. Backtesting je rozdělen do 5. fází, který je velice unikátní. První fáze spočívá v identifikaci vhodných klouzavých průměru, další fáze identifikovává nejvhodnější oscilátory, poté se kontrolují možné svíčkové formace a následně se kontrolují nejběžnější pravidla. Jako posledně příjde poslední 5. fáze, která postupně vytváří algoritmy, které jsou založeny na datech z přecházející iterace. Výsledný algoritmus je poté lépe schopen identifikovat vstupy a výstupy do možných obchodů.

6.7 OBCHODOVÁNÍ V REALNÉM ČASE

OBRÁZEK 16 - TRADINGVIEW - 23.3. 2023 - NYSE:BOXD

Ve dni 23.3. 2023 byla naše aplikace vyzkoušena v reálném čase. Jednalo se o akcii NYSE: BOXD (na Newyorské burze). Náš algoritmus ji zaznamenal, protože měla nějvětší nárust objemu obchodování z pohledu celého trhu. Následně algoritmus byl aktivován (ručně) a skript začal posílat objednávky do IB klientu. Je znát, že program zabírá obrovské množství paměti a CPU výkonu i místa na disku. Odezva byla poměrně vysoká, většina obchodů se neaktivovala (skoro 24 obchodů nebylo aktivováno). Tento projekt mohl využívat až $5000. Každý obchod však byl nastaven na fixní množství 2000 akcií. Poplatek za každou objednávku (vyhotovenou) byl $1.8 (nákup i prodej). Algoritmus byl schopen vyprodukovat při počtu 11 obchodů (vysoké množství vůči takové společnosti) cca $53 – to bylo způsobeno tím, že se jednalo o perfektní nastavení a podmínky pro náš algoritmus. Avšak 54.55 % obchodů bylo profitových. Nejvyšší ztráta byla $86 a nevyšší profit byl $47. Drawdown byl $94 taktéž při nejvyšší volatilitě. Používali jsme Higher highs svíčkovou formaci jako základ (hlavní indikátor), pravidla pro obchod objem*cena_akcie vyšší jak $100000 za posledních 5minut (druhý nejdůležitější indikátor). Byly použity všechny oscilátory a klouzavé průměry, avšak každý s jinou váhou. Obchodovalo se s daty, které se aktualizovávali každou minutu. Průměrný profit za každý obchod $4,82.

TABULKA 6 - OBCHODY

#	Entry Time	Exit Time	Entry Price	Stop Loss	Profit Taker	Exit Price
1	2023-03-23 04:06:00	2023-03-23 04:16:00	0.2133	0.2046	0.2299	0.2168
2	2023-03-23 04:12:00	2023-03-23 04:17:00	0.2182	0.2101	0.2356	0.2131
3	2023-03-23 04:18:00	2023-03-23 04:27:00	0.2153	0.2067	0.2307	0.2161
4	2023-03-23 04:22:00	2023-03-23 04:32:00	0.2158	0.2074	0.2318	0.2072
5	2023-03-23 04:28:00	2023-03-23 04:38:00	0.2098	0.2019	0.2245	0.2124
6	2023-03-23 04:33:00	2023-03-23 04:46:00	0.2129	0.2048	0.2276	0.2163
7	2023-03-23 04:41:00	2023-03-23 04:52:00	0.2139	0.2054	0.2284	0.2164
8	2023-03-23 04:47:00	2023-03-23 04:57:00	0.2167	0.2079	0.2311	0.2142
9	2023-03-23 04:54:00	2023-03-23 05:01:00	0.2152	0.2063	0.2296	0.2186

#	Entry Time	Exit Time	Entry Price	Stop Loss	Profit Taker	Exit Price
10	2023-03-23 05:03:00	2023-03-23 05:11:00	0.2186	0.2097	0.2329	0.2148
11	2023-03-23 05:08:00	2023-03-23 05:19:00	0.2144	0.2058	0.2289	0.2191

7 ZÁVĚR

Závěr této diplomové práce ukazuje, že byly použity kvalitní data a přísné postupy pro analýzu dat s cílem vytvořit algoritmus pro intradenní obchodování s akciemi. Algoritmus byl navržen tak, aby reagoval na dění na trhu a využíval data za poslední 3 roky k definování nejlepších pravidel, oscilátorů a klouzavých průměrů pro každý den. Výsledkem je algoritmus, který se dá použít v reálném fungování, ale je extrémně náročný na CPU, RAM a úložiště.

Důležitým faktorem pro úspěšnost obchodní strategie pro intradenní momentum obchodování s akciemi je volba správných pravidel pro obchodování s akciemi. Backtestování bylo úspěšné a umožnilo identifikovat nejvhodnější průměrné hodnoty, oscilátory a pravidla pro obchodování s akciemi. Vylepšený algoritmus pro detekci svíčkových formací a generování signálů nákupu a prodeje se ukázal jako účinný a zlepšil výkonnost obchodní strategie.

Celkově lze tedy říci, že diplomová práce úspěšně dosáhla svého cíle vytvořit algoritmus pro intradenní obchodování s akciemi a provedla řadu backtestů, které pomohly identifikovat nejvhodnější pravidla pro obchodování s akciemi.

8 LITERATURA

A universal performance measure. **Keating, C., & Shadwick, W. F. 2002.** místo neznámé : Journal of Performance Measurement, 2002.

Achelis, S. B. 2000. *Technical analysis from A to Z: Covers every trading tool from the absolute breadth index to the zig zag.* místo neznámé : McGraw-Hill., 2000. ISBN-10: 1557388164.

—. 2000. *Technical analysis from A to Z: Covers every trading tool from the absolute breadth index to the zig zag.* místo neznámé : McGraw-Hill., 2000. ISBN-10: 1557388164.

Aldridge, I. 2010. *High-frequency trading: A practical guide to algorithmic strategies and trading systems.* místo neznámé : John Wiley & Sons., 2010. ISBN-10: 0470563761.

Algorithmic trading and the market for liquidity. **Hendershott, T., & Riordan, R. 2013.** místo neznámé : Journal of Financial and Quantitative Analysis,, 2013, Sv. 4, stránky 1001-1024.

Almgren, R., & Chriss, N. 2005. *Optimal execution of portfolio transactions.* místo neznámé : Journal of Risk, 2005.

Appel, G., & Hitschler, F. 1988. *Stock market trading systems.* místo neznámé : Dow Jones-Irwin., 1988. ISBN-10: 087094195X.

Automated news reading: Stock price prediction based on financial news using context-capturing features. **Hagenau, M., Liebmann, M., & Neumann, D. 2013.** místo neznámé : Decision Support Systems, 2013, Sv. 3, stránky 685-697.

Avellaneda, M., & Lee, J. H. 2010. *Statistical arbitrage in the US equities market.* místo neznámé : Quantitative Finance, 2010. stránky 761-782. Sv. 7. ISBN-10: 9811239495.

Bailey, D. H., Borwein, J. M., López de Prado, M., & Zhu, Q. J. 2014. *Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance.* místo neznámé : Notices of the AMS, 2014. stránky 458-471. Sv. 5.

Bodie, Z., Kane, A., & Marcus, A. J. 2014. *Investments.* místo neznámé : McGraw-Hill., 2014. ISBN-10: 1260571157.

Bollinger, J. 2001. *Bollinger on Bollinger Bands.* místo neznámé : McGraw-Hill., 2001. 9780071373685.

Brown, C. 2008. *Technical analysis for the trading professional: Strategies and techniques for today's turbulent financial markets.* místo neznámé : McGraw Hill Professional., 2008. 007175914X.

Bulkowski, T. N. 2005. *Encyclopedia of chart patterns.* místo neznámé : John Wiley & Sons., 2005. ISBN-10: 1119739683.

Candlestick technical trading strategies: Can they create value for investors? **Marshall, B. R., Young, M. R., & Rose, L. C. 2006.** místo neznámé : Journal of Banking & Finance, 2006.

Cartea, Álvaro, Jaimungal, Sebastian a Penalva, José. 2015. *Algorithmic and high-frequency trading.* místo neznámé : Cambridge : Cambridge University Press, 2015. 978-1-107-09114-6..

Classification-based financial markets prediction using deep neural networks. **Dixon, M., Klabjan, D., & Bang, J. H. 2020.** místo neznámé : Algorithmic Finance, 2020.

Dynamic neural networks in high-frequency trading. **Galeshchuk, S., & Mukhina, I. 2016.** místo neznámé : Journal of Asset Management, 2016, Sv. 5, stránky 339-349.

Equilibrium fast trading. **Biais, B., Foucault, T., & Moinas, S. 2015.** místo neznámé : Journal of Financial Economics, 2015.

Ernest, Chan. 2017. *Machine Trading: Deploying Computer Algorithms to Conquer the Markets. 1.* místo neznámé : Hoboken : John Wiley & Sons, Inc.,, 2017. 978-1-119-21960-6..

—. 2021. *Quantitative Trading: How to Build Your Own Algorithmic Trading Business. 2.* místo neznámé : Hoboken : John Wiley & Sons, Inc.,, 2021. 978-1-119-80006-4..

Fisher, R. A. 1925. *Statistical methods for research workers.* místo neznámé : Oliver and Boyd., 1925. ISBN-10: 9351286584.

Gomber, P., Arndt, B., Lutat, M., & Uhle, T. 2011. *High-frequency trading.* místo neznámé : Goethe University Frankfurt, Chair of e-Finance., 2011.

Granville, J. E. 1963. *Granville's New Key to Stock Market Profit.* místo neznámé : Prentice-Hall, 1963. ISBN-10: 125802649X.

High-frequency trading. **Gomber, P., Arndt, B., Lutat, G., & Uhle, T. 2011.** místo neznámé : Goethe University Frankfurt, Chair of e-Finance, 2011.

High-frequency trading. Business & Information Systems Engineering. **Gomber, P., Arndt, B., Lutat, M., & Uhle, T. 2011.** 2011.

How to rate management of investment funds. **Treynor, J. L. 1965.** místo neznámé : Harvard Business Review, 1965.

Hull, J. C. 2013. *Options, futures, and other derivatives.* místo neznámé : Pearson Education., 2013. ISBN-10: 1292410655.

Chan, E. P. 2008. *Quantitative trading: How to build your own algorithmic trading business.* místo neznámé : John Wiley & Sons., 2008. ISBN-10: 1119800064.

Is technical analysis in the foreign exchange market profitable? A genetic programming approach. **Neely, C. J., Weller, P. A.,& Dittmar, R. 1997.** místo neznámé : Journal of Financial and Quantitative Analysis, 1997.

Kaufman, P. J. 2013. *Trading systems and methods.* místo neznámé : John Wiley & Sons., 2013. ISBN-10: 1119605350.

—. 2013. *Trading systems and methods.* místo neznámé : John Wiley & Sons., 2013. ISBN-10: 1119605350.

Kirkpatrick, C. D., & Dahlquist, J. R. 2010. *Technical analysis: The complete resource for financial market technicians.* místo neznámé : FT Press., 2010. ISBN-10: 0134137043.

—. 2010. *Technical analysis: The complete resource for financial market technicians.* místo neznámé : FT Press., 2010. ISBN-10: 0134137043.

Lane's stochastic. **Lane, G. C. 1984.** místo neznámé : Technical Analysis of Stocks & Commodities, 1984.

Learning the art of the science of trading: The Sterling ratio. **Young, M. R. 1991.** místo neznámé : The Technical Analyst, 1991.

Leinweber, D. J. 2009. *Nerds on Wall Street: Math, machines, and wired markets.* místo neznámé : John Wiley & Sons, 2009. ISBN-10: 0471369462.

Low-latency trading. **Hasbrouck, J., & Saar, G. 2013.** místo neznámé : Journal of Financial Markets, 2013.

Machine learning in finance: The case of deep learning for option pricing. **Dixon, M., Klabjan, D., & Bang, J. H. 2020.** místo neznámé : Journal of Investment Management, 2020, Sv. 2, stránky 1-20.

Market liquidity and funding liquidity. **Brunnermeier, M. K., & Pedersen, L. H. 2009.** místo neznámé : The Review of Financial Studies, 2009.

Momentum strategies: Evidence from the Pacific Basin stock markets. **Moser, G. P. 2012.** místo neznámé : International Journal of Financial Research, 2012, Sv. 2, stránky 28-46.

Morris, G. L. 2012. *Candlestick charting explained: Timeless techniques for trading stocks and futures.* místo neznámé : McGraw Hill Professional, 2012. ISBN-10: 007146154X.

Murphy, J. J. 2012. *Trading with intermarket analysis: A visual approach to beating the financial markets using exchange-traded funds.* místo neznámé : John Wiley & Sons., 2012. ISBN-10: 1119210011.

Mutual fund performance. **Sharpe, W. F. 1966.** místo neznámé : Journal of Business, 1966.

New concepts in technical trading systems. **Wilder, J. W. 1978.** místo neznámé : Trend Research, 1978.

Nison, S. 1991. *Japanese candlestick charting techniques: A contemporary guide to the ancient investment techniques of the Far East.* místo neznámé : Prentice Hall Press, 1991. ISBN-10: 0139316507.

Pardo, R. 2008. *The evaluation and optimization of trading strategies.* místo neznámé : John Wiley & Sons, 2008. ISBN-10: 0470128011.

Performance measurement in a downside risk framework. **Sortino, F. A., & Price, L. N. 1994.** místo neznámé : Journal of Investing, 1994.

Pring, M. J. 2002. *Technical analysis explained: The successful investor's guide to spotting investment trends and turning points.* místo neznámé : McGraw Hill Professional., 2002. ISBN-10: 0071825177.

Rise of the machines: Algorithmic trading in the foreign exchange market. **Chaboud, A. P., Chiquoine, B., Hjalmarsson, E., & Vega, C. 2014.** místo neznámé : The Journal of Finance, 2014, Sv. 5, stránky 2045-2084.

Schwager, J. D. 1996. *Technical analysis: Study guide.* místo neznámé : John Wiley & Sons., 1996. ISBN-10: 0735200653.

Student. 1908. *The probable error of a mean.* místo neznámé : Biometrika, 1908. 978-0-387-94039-7.

Support and resistance: Trading by reading a market. **Suri, D. 2015.** místo neznámé : Futures Magazine., 2015.

Tapscott, D., & Tapscott, A. 2016. *Blockchain revolution: How the technology behind Bitcoin is changing money, business, and the world.* místo neznámé : Penguin, 2016. ISBN-10: 1101980133.

The adaptive markets hypothesis: Market efficiency from an evolutionary perspective. **Lo, A. W. 2004.** místo neznámé : Journal of Portfolio Management, 2004.

The diversity of high-frequency traders. **Hagströmer, B., & Norden, L. 2013.** místo neznámé : Journal of Financial Markets, 2013.

The information ratio. **Goodwin, T. H. 1998.** místo neznámé : Financial Analysts Journal, 1998.

The performance of mutual funds in the period 1945-1964. **Jensen, M. C. 1968.** místo neznámé : The Journal of Finance, 1968.

The science of algorithmic trading and portfolio management. **Kissell, R. 2013.** místo neznámé : Academic Press, 2013.

Universal features of price formation in financial markets: Perspectives from deep learning. **Sirignano, J., & Cont, R. 2019.** místo neznámé : Quantitative Finance, 2019, Sv. 9, stránky 1449-1459.

Volatility clustering in financial markets: Empirical facts and agent-based models. **Cont, R. 2007.** místo neznámé : Springer, 2007.

What do we know about high-frequency trading? **Jones, C. M. 2013.** místo neznámé : Columbia Business School Research Paper, 2013.

What happened to the quants in August 2007? Evidence from factors and transactions data. **Khandani, A. E., & Lo, A. W. 2007.** místo neznámé : Journal of Investment Management, 2007.

9 SEZNAM OBRÁZKŮ A TABULEK

9.1 SEZNAM OBRÁZKŮ

Obrázek 1 - csv soubory v programu 36
Obrázek 2 - kalkulace indikátorů 47
Obrázek 3 - kalkulace indikátorů 48
Obrázek 4 - kalkulace f testu 50
Obrázek 5 - kalkulace Beta & Jensens alpha 50
Obrázek 6 - příklad kodu - gui.py 51
Obrázek 7 GUI rozložení 52
Obrázek 8 - gui rozložení při zapnutí screeneru 52
Obrázek 9 - list nejobchodovanějších akcií + vlastní výběr 53
Obrázek 10 - ib aktivní/ provedené objednávky 53
Obrázek 11 - obchodovancí klient – v chodu 54
Obrázek 12 – obchodovací KLIENT – PAUZNUTÍ 54
Obrázek 13 - obchodovací KLIENT – PŘED startem 54
Obrázek 14 - ib backtesting – kousek kodu ..... **Chyba! Záložka není definována.**
Obrázek 15 – strategie 56
Obrázek 16 - TradingView - 23.3. 2023 - NYSE:BOXD 65
Obrázek 17 - Nepovedený test 77
Obrázek 18 - Algoritmus #1 77
Obrázek 19 - Algoritmus #2 78
Obrázek 20 - Trading client #1 79
Obrázek 21 - Trading client #2 80
Obrázek 22 - Trading client #3 81
Obrázek 23 - struktura složky 82
Obrázek 24 - Output statistika 83
Obrázek 25 - Output statistika 83
Obrázek 26 - Statistická analýza #1 84
Obrázek 27 - Statistická analýza #2 85

Obrázek 28 - Statistická analýza #3 86
Obrázek 29 - Statistická analýza #4 87
Obrázek 30 Statistická analýza #5 88
Obrázek 31 - Statistická analýza #6 89
Obrázek 32 - Screener #1 90
Obrázek 33 - Screener #2 91
Obrázek 34 - Screener #3 92
Obrázek 35 - Screener #4 93
Obrázek 36 - Screener #5 94
Obrázek 37 - screener #6 95
Obrázek 38 - Screener #7 96

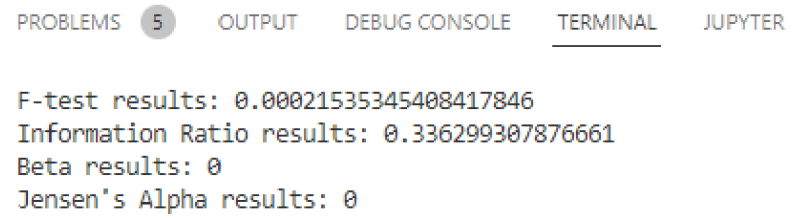
9.2 SEZNAM TABULEK

Tabulka 1 – první backtest 57
Tabulka 2 - druhý backtest 59
Tabulka 3 - třetí backtest 60
Tabulka 4 - čtvrtý backtest 62
Tabulka 5 - páty backtest 63
Tabulka 6 - obchody 66

10 PŘÍLOHY

```
        self.stock_list = ttk.Treeview(self, columns=("Stocks"), show="headings")
        self.stock_list.heading("Stocks", text="Stocks")
        self.stock_list.pack(fill="both", expand=True)

        bottom_frame = ttk.Frame(self)
        bottom_frame.pack(side="bottom", fill="x")

        self.start_button = ttk.Button(bottom_frame, text="Start Trading", command=self.start_trading)
        self.start_button.pack(side="left")

        self.pause_button = ttk.Button(bottom_frame, text="Pause Trading", command=self.pause_trading, state="disabled")
        self.pause_button.pack(side="left")

        self.resume_button = ttk.Button(bottom_frame, text="Resume Trading", command=self.resume_trading, state="disabled")
        self.resume_button.pack(side="left")

        self.stop_button = ttk.Button(bottom_frame, text="Stop Trading", command=self.stop_trading, state="disabled")
        self.stop_button.pack(side="left")

    def add_stock(self):
        stock = self.stock_entry.get()
        self.stock_entry.delete(0, "end")
        self.algorithm.add_stock(stock)
        self.stock_list.insert("", "end", values=(stock,))

    def start_trading(self):
        self.algorithm.start()
        self.start_button.config(state="disabled")
        self.pause_button.config(state="normal")
        self.stop_button.config(state="normal")

    def pause_trading(self):
        self.algorithm.pause()
        self.pause_button.config(state="disabled")
        self.resume_button.config(state="normal")

    def resume_trading(self):
        self.algorithm.resume()
        self.resume_button.config(state="disabled")
        self.pause_button.config(state="normal")

    def stop_trading(self):
        self.algorithm.stop()
        self.start_button.config(state="normal")
        self.pause_button.config(state="disabled")
        self.resume_button.config(state="disabled")
        self.stop_button.config(state="disabled")
```

Obrázek 17 - stocklist



OBRÁZEK 18 - NEPOVEDENÝ TEST

```python
from alpha_vantage.timeseries import TimeSeries
from ib_insync import IB
from ib_insync import IB, Stock, MarketOrder
import threading
import time

class Algorithm:
    def __init__(self, alpha_vantage_api_key, ib_host='127.0.0.1', ib_port=7497):
        self.ts = TimeSeries(key=alpha_vantage_api_key, output_format='pandas')
        self.stocks = []
        self.strategy = 'Higher Highs'
        self.ib = IB()
        self.ib.connect(ib_host, ib_port, clientId=1)
        self.trading = False
        self.paused = False


    def set_strategy(self, strategy):
        self.strategy = strategy

    def add_stock(self, stock):
        self.stocks.append(stock)

    def start(self):
        self.trading = True
        self.paused = False
        trading_thread = threading.Thread(target=self.run_trading)
        trading_thread.start()

    def pause(self):
        self.paused = True

    def resume(self):
        self.paused = False

    def stop(self):
        self.trading = False

    def run_trading(self):
        while self.trading:
            if not self.paused:
                for stock in self.stocks:
                    # Fetch data from AlphaVantage
                    data, _ = self.ts.get_intraday(symbol=stock, interval='5min', outputsize='compact')
                    # Implement your trading strategy here
                    signal = self.evaluate_signal(stock, data)
```

OBRÁZEK 19 - ALGORITMUS #1

```
                if signal == 'BUY':
                    contract = Stock(stock, 'SMART', 'USD')
                    order = MarketOrder('BUY', 10)  # Replace 10 with the desired number of shares
                    trade = self.ib.placeOrder(contract, order)
                    self.ib.sleep(1)  # Give the order some time to be executed

            time.sleep(5)  # Pause for 5 seconds before checking again

    def evaluate_signal(self, stock, data):
        return 'BUY'
```

OBRÁZEK 20 - ALGORITMUS #2

```
class TradingClient(ttk.Frame):
    def __init__(self, master=None, **kw):
        super().__init__(master, **kw)
        self.master = master
        self.master.title("Trading Client")
        self.pack(fill="both", expand=True)

        self.algorithm = Algorithm(alpha_vantage_api_key='JACJKLH87O6AUMBM')


        self.create_widgets()

    def create_widgets(self):
        top_frame = ttk.Frame(self)
        top_frame.pack(side="top", fill="x")

        self.strategy_var = tk.StringVar()
        self.strategy_var.set("Higher Highs")
        ttk.Label(top_frame, text="Strategy:").pack(side="left")
        self.strategy_menu = ttk.OptionMenu(top_frame, self.strategy_var, "Higher Highs", "Higher Highs", command=self.algorithm.set_strategy)
        self.strategy_menu.pack(side="left")

        ttk.Label(top_frame, text="Add Stock:").pack(side="left")
        self.stock_entry = ttk.Entry(top_frame)
        self.stock_entry.pack(side="left")

        self.add_stock_button = ttk.Button(top_frame, text="Add", command=self.add_stock)
        self.add_stock_button.pack(side="left")

        self.stock_list = ttk.Treeview(self, columns=("Stocks"), show="headings")
        self.stock_list.heading("Stocks", text="Stocks")
        self.stock_list.pack(fill="both", expand=True)

        bottom_frame = ttk.Frame(self)
        bottom_frame.pack(side="bottom", fill="x")

        self.start_button = ttk.Button(bottom_frame, text="Start Trading", command=self.start_trading)
        self.start_button.pack(side="left")

        self.pause_button = ttk.Button(bottom_frame, text="Pause Trading", command=self.pause_trading, state="disabled")
        self.pause_button.pack(side="left")

        self.resume_button = ttk.Button(bottom_frame, text="Resume Trading", command=self.resume_trading, state="disabled")
```

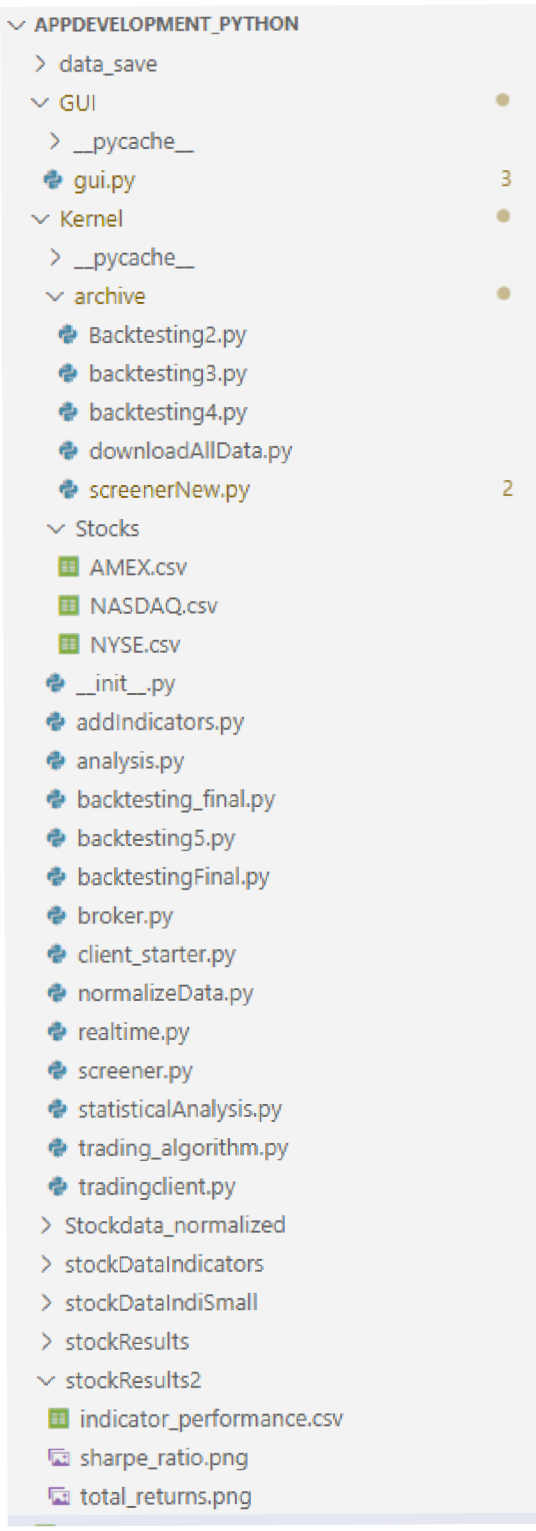
OBRÁZEK 21 - TRADING CLIENT #1

```
        self.stock_list = ttk.Treeview(self, columns=("Stocks"), show="headings")
        self.stock_list.heading("Stocks", text="Stocks")
        self.stock_list.pack(fill="both", expand=True)

        bottom_frame = ttk.Frame(self)
        bottom_frame.pack(side="bottom", fill="x")

        self.start_button = ttk.Button(bottom_frame, text="Start Trading", command=self.start_trading)
        self.start_button.pack(side="left")

        self.pause_button = ttk.Button(bottom_frame, text="Pause Trading", command=self.pause_trading, state="disabled")
        self.pause_button.pack(side="left")

        self.resume_button = ttk.Button(bottom_frame, text="Resume Trading", command=self.resume_trading, state="disabled")
        self.resume_button.pack(side="left")

        self.stop_button = ttk.Button(bottom_frame, text="Stop Trading", command=self.stop_trading, state="disabled")
        self.stop_button.pack(side="left")

    def add_stock(self):
        stock = self.stock_entry.get()
        self.stock_entry.delete(0, "end")
        self.algorithm.add_stock(stock)
        self.stock_list.insert("", "end", values=(stock,))

    def start_trading(self):
        self.algorithm.start()
        self.start_button.config(state="disabled")
        self.pause_button.config(state="normal")
        self.stop_button.config(state="normal")

    def pause_trading(self):
        self.algorithm.pause()
        self.pause_button.config(state="disabled")
        self.resume_button.config(state="normal")

    def resume_trading(self):
        self.algorithm.resume()
        self.resume_button.config(state="disabled")
        self.pause_button.config(state="normal")

    def stop_trading(self):
        self.algorithm.stop()
        self.start_button.config(state="normal")
        self.pause_button.config(state="disabled")
        self.resume_button.config(state="disabled")
        self.stop_button.config(state="disabled")
```

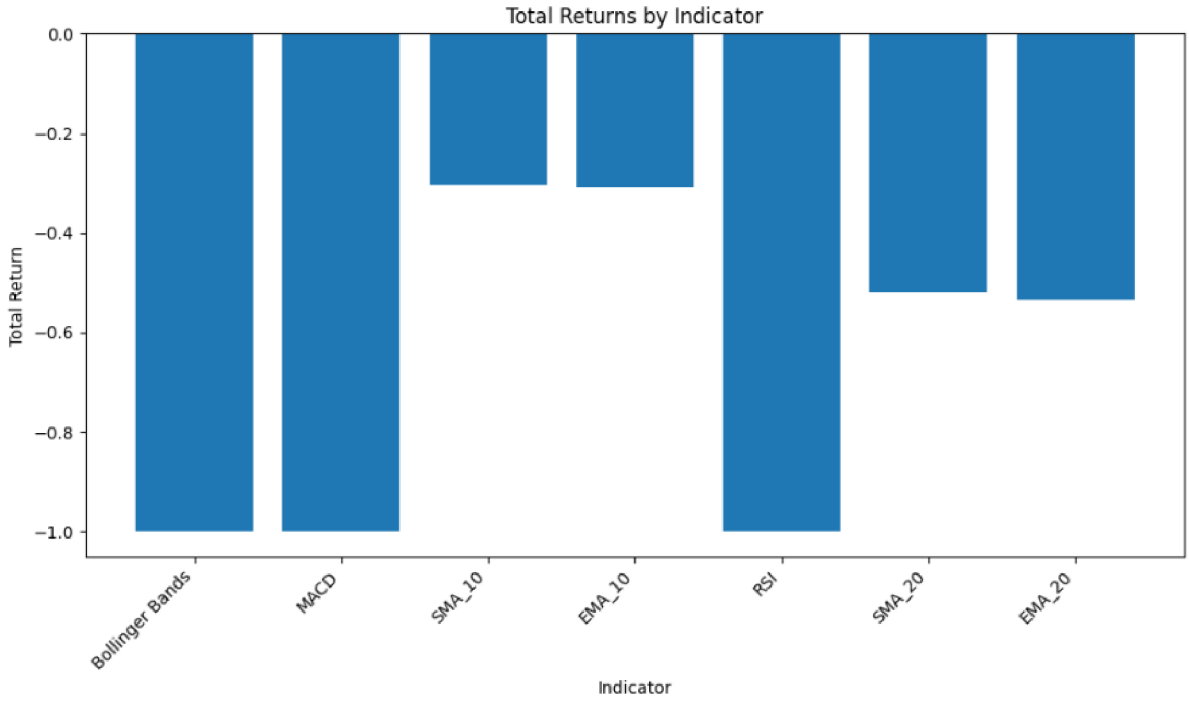
OBRÁZEK 22 - TRADING CLIENT #2

```
class TradingClient(ttk.Frame):
    def __init__(self, master=None, **kw):
        super().__init__(master, **kw)
        self.master = master
        self.master.title("Trading Client")
        self.pack(fill="both", expand=True)

        self.algorithm = Algorithm(alpha_vantage_api_key='JACJKLH87O6AUMBM')


        self.create_widgets()

    def create_widgets(self):
        top_frame = ttk.Frame(self)
        top_frame.pack(side="top", fill="x")

        self.strategy_var = tk.StringVar()
        self.strategy_var.set("Higher Highs")
        ttk.Label(top_frame, text="Strategy:").pack(side="left")
        self.strategy_menu = ttk.OptionMenu(top_frame, self.strategy_var, "Higher Highs", "Higher Highs", command=self.algorithm.set_strategy)
        self.strategy_menu.pack(side="left")

        ttk.Label(top_frame, text="Add Stock:").pack(side="left")
        self.stock_entry = ttk.Entry(top_frame)
        self.stock_entry.pack(side="left")

        self.add_stock_button = ttk.Button(top_frame, text="Add", command=self.add_stock)
        self.add_stock_button.pack(side="left")

        self.stock_list = ttk.Treeview(self, columns=("Stocks"), show="headings")
        self.stock_list.heading("Stocks", text="Stocks")
        self.stock_list.pack(fill="both", expand=True)

        bottom_frame = ttk.Frame(self)
        bottom_frame.pack(side="bottom", fill="x")

        self.start_button = ttk.Button(bottom_frame, text="Start Trading", command=self.start_trading)
        self.start_button.pack(side="left")

        self.pause_button = ttk.Button(bottom_frame, text="Pause Trading", command=self.pause_trading, state="disabled")
        self.pause_button.pack(side="left")

        self.resume_button = ttk.Button(bottom_frame, text="Resume Trading", command=self.resume_trading, state="disabled")
```

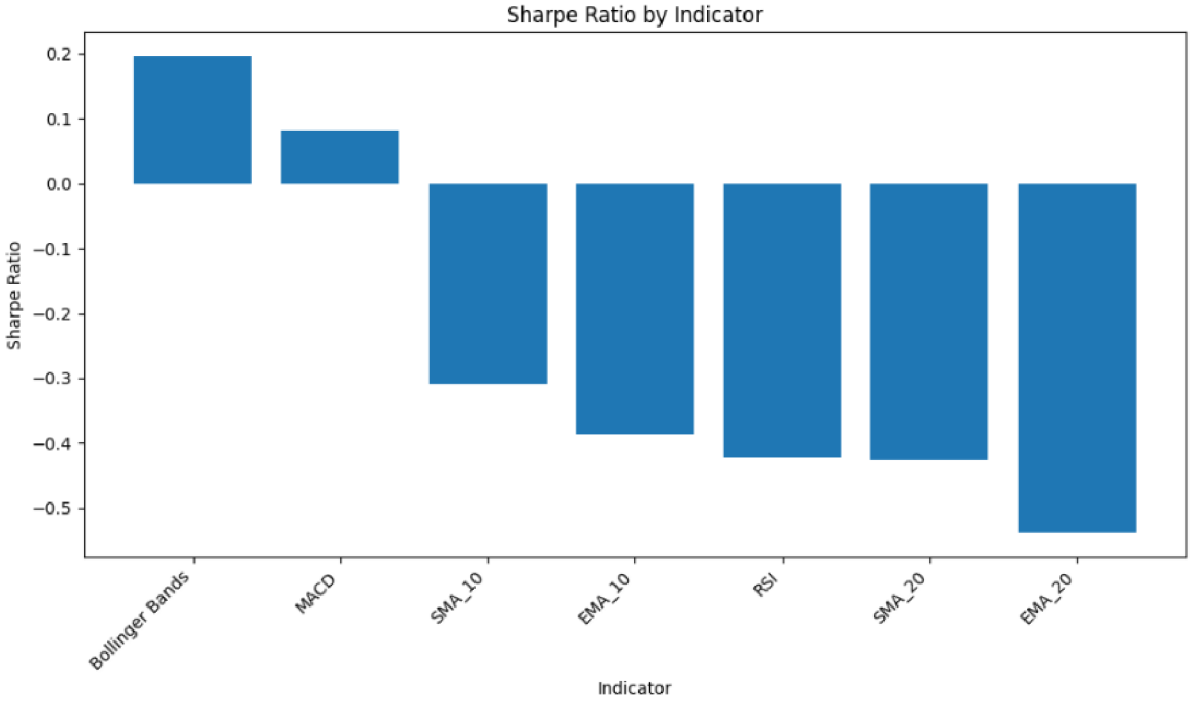
OBRÁZEK 23 - TRADING CLIENT #3



OBRÁZEK 24 - STRUKTURA SLOŽKY



OBRÁZEK 25 - OUTPUT STATISTIKA



OBRÁZEK 26 - OUTPUT STATISTIKA

```
import os
import glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind



def analyze_stock(stock_df):
    # Calculate the daily returns
    try:
        stock_df['daily_return'] = stock_df['close'].pct_change()
    except ZeroDivisionError:
        print("Error: Cannot calculate percentage change. Possible division by zero.")
        return None

    # Calculate the log returns
    try:
        stock_df['log_return'] = np.log(stock_df['close']).diff()
    except ZeroDivisionError:
        print("Error: Cannot calculate log returns. Possible division by zero.")
        return None

    # Calculate the moving averages
    try:
        stock_df['sma_10'] = stock_df['close'].rolling(window=10).mean()
        stock_df['sma_20'] = stock_df['close'].rolling(window=20).mean()
        stock_df['ema_10'] = stock_df['close'].ewm(span=10).mean()
        stock_df['ema_20'] = stock_df['close'].ewm(span=20).mean()
    except ZeroDivisionError:
        print("Error: Cannot calculate moving averages. Possible division by zero.")
        return None

    # Calculate the Bollinger Bands
    try:
        rolling_mean = stock_df['close'].rolling(window=20).mean()
        rolling_std = stock_df['close'].rolling(window=20).std()
        stock_df['bb_upper'] = rolling_mean + 2 * rolling_std
        stock_df['bb_lower'] = rolling_mean - 2 * rolling_std
    except ZeroDivisionError:
        print("Error: Cannot calculate Bollinger Bands. Possible division by zero.")
        return None

    # Calculate the Relative Strength Index (RSI)
    try:
        delta = stock_df['close'].diff()
        gain = delta.where(delta > 0, 0)
        loss = -delta.where(delta < 0, 0)
```

OBRÁZEK 27 - STATISTICKÁ ANALÝZA #1

```
    except ZeroDivisionError:
        print("Error: Cannot calculate Bollinger Bands. Possible division by zero.")
        return None

    # Calculate the Relative Strength Index (RSI)
    try:
        delta = stock_df['close'].diff()
        gain = delta.where(delta > 0, 0)
        loss = -delta.where(delta < 0, 0)
        avg_gain = gain.rolling(window=14).mean()
        avg_loss = loss.rolling(window=14).mean()
        rs = avg_gain / avg_loss
        stock_df['rsi'] = 100 - (100 / (1 + rs))
    except ZeroDivisionError:
        print("Error: Cannot calculate RSI. Possible division by zero.")
        return None

    # Calculate the Moving Average Convergence Divergence (MACD)
    try:
        exp1 = stock_df['close'].ewm(span=12, adjust=False).mean()
        exp2 = stock_df['close'].ewm(span=26, adjust=False).mean()
        stock_df['macd'] = exp1 - exp2
        stock_df['signal'] = stock_df['macd'].ewm(span=9, adjust=False).mean()
    except ZeroDivisionError:
        print("Error: Cannot calculate MACD. Possible division by zero.")
        return None

    # Calculate the On-Balance Volume (OBV)
    try:
        stock_df['obv'] = np.where(stock_df['daily_return'] > 0, stock_df['volume'], -stock_df['volume']).cumsum()
    except ZeroDivisionError:
        print("Error: Cannot calculate OBV. Possible division by zero.")
        return None

# Perform a t-test on the daily returns to test for significance
    try:
        mean_return = stock_df['daily_return'].mean()
        t_stat, p_value = ttest_ind(stock_df['daily_return'], np.zeros_like(stock_df['daily_return']))
        is_significant = p_value < 0.05
    except ZeroDivisionError:
        print("Error: Cannot perform t-test. Possible division by zero.")
        return None
```

OBRÁZEK 28 - STATISTICKÁ ANALÝZA #2

```
    # Return a dictionary of results
    results = {
        'mean_return': mean_return,
        't_stat': t_stat,
        'p_value': p_value,
        'is_significant': is_significant
    }

    return results

def analyze_stocks(input_folder, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    file_paths = glob.glob(f"{input_folder}/*_5min_normalized_indicators.csv")

    results_list = []
    performance = pd.DataFrame(columns=['Indicator', 'Sharpe Ratio', 'Total Return', 'Annualized Return', 'Max Drawdown'])


    for file_path in file_paths:
        stock_name = os.path.basename(file_path).split("_")[0]

        df = pd.read_csv(file_path)
        df['timestamp'] = pd.to_datetime(df['timestamp'])
        df.set_index('timestamp', inplace=True)


        # Filter rows where the product of volume and price is greater than 10,000
        df = df[df['volume'] * df['close'] > 10000]

        if len(df) > 0:
            stock_results = analyze_stock(df)
            # ...
        else:
            print(f"Not enough data for stock {stock_name} after filtering. Skipping...")
            continue

        if stock_results is None:
            print(f"Analysis for stock {stock_name} failed. Skipping...")
            continue

        stock_results['stock_name'] = stock_name
        results_list.append(stock_results)
```

OBRÁZEK 29 - STATISTICKÁ ANALÝZA #3

```python
# Compute the Sharpe Ratio for each indicator
performance = pd.DataFrame(columns=['Indicator', 'Sharpe Ratio', 'Total Return', 'Annualized Return', 'Max Drawdown'])

for indicator in ['sma_10', 'sma_20', 'ema_10', 'ema_20', 'macd']:
    try:
        indicator_returns = df['close'] / df[indicator] - 1
        indicator_returns.fillna(0, inplace=True)
        indicator_sharpe_ratio = indicator_returns.mean() / indicator_returns.std() * np.sqrt(252)
        indicator_total_return = (indicator_returns + 1).cumprod().iloc[-1] - 1
        indicator_annualized_return = (1 + indicator_total_return) ** (252 / len(df)) - 1
        indicator_max_drawdown = (indicator_returns + 1).cumprod().div((indicator_returns + 1).cumprod().cummax()).sub(1).min()

        performance = performance.append({'Indicator': indicator.upper(), 'Sharpe Ratio': indicator_sharpe_ratio,
                                          'Total Return': indicator_total_return,
                                          'Annualized Return': indicator_annualized_return,
                                          'Max Drawdown': indicator_max_drawdown}, ignore_index=True)
    except ZeroDivisionError:
        print(f"Error: Cannot calculate performance for {indicator.upper()}. Possible division by zero.")

# Compute the Sharpe Ratio for the Bollinger Bands strategy
try:
    rolling_mean = df['close'].rolling(window=20).mean()
    rolling_std = df['close'].rolling(window=20).std()
    df['bb_upper'] = rolling_mean + 2 * rolling_std
    df['bb_lower'] = rolling_mean - 2 * rolling_std

    bb_returns = np.where(df['close'] > df['bb_upper'], -1, np.nan)
    bb_returns = np.where(df['close'] < df['bb_lower'], 1, bb_returns)
    bb_returns = pd.Series(bb_returns).fillna(0)
    bb_sharpe_ratio = bb_returns.mean() / bb_returns.std() * np.sqrt(252)
    bb_total_return = (bb_returns + 1).cumprod().iloc[-1] - 1
    bb_annualized_return = (1 + bb_total_return) ** (252 / len(df)) - 1
    bb_max_drawdown = (bb_returns + 1).cumprod().div((bb_returns + 1).cumprod().cummax()).sub(1).min()

    performance = performance.append({'Indicator': 'Bollinger Bands', 'Sharpe Ratio': bb_sharpe_ratio,
                                      'Total Return': bb_total_return,
                                      'Annualized Return': bb_annualized_return,
                                      'Max Drawdown': bb_max_drawdown}, ignore_index=True)
except ZeroDivisionError:
    print("Error: Cannot calculate performance for Bollinger Bands. Possible division by zero.")
```

OBRÁZEK 30 - STATISTICKÁ ANALÝZA #4

```
    # Compute the Sharpe Ratio for the RSI strategy
    rsi_returns = np.where(df['rsi'] > 70, -1, np.nan)
    rsi_returns = np.where(df['rsi'] < 30, 1, rsi_returns)
    rsi_returns = pd.Series(rsi_returns).fillna(0)
    rsi_sharpe_ratio = rsi_returns.mean() / rsi_returns.std() * np.sqrt(252)
    rsi_total_return = (rsi_returns + 1).cumprod().iloc[-1] - 1
    rsi_annualized_return = (1 + rsi_total_return) ** (252 / len(df)) - 1
    rsi_max_drawdown = (rsi_returns + 1).cumprod().div((rsi_returns + 1).cumprod().cummax()).sub(1).min()

    performance = performance.append({'Indicator': 'RSI', 'Sharpe Ratio': rsi_sharpe_ratio,
                                      'Total Return': rsi_total_return, 'Annualized Return': rsi_annualized_return,
                                      'Max Drawdown': rsi_max_drawdown}, ignore_index=True)

    # Compute the Sharpe Ratio for the MACD strategy
    macd_returns = np.where(df['macd'] > df['signal'], -1, np.nan)
    macd_returns = np.where(df['macd'] < df['signal'], 1, macd_returns)
    macd_returns = pd.Series(macd_returns).fillna(0)

    macd_sharpe_ratio = macd_returns.mean() / macd_returns.std() * np.sqrt(252)
    macd_total_return = (macd_returns + 1).cumprod().iloc[-1] - 1
    macd_annualized_return = (1 + macd_total_return) ** (252 / len(df)) - 1
    macd_max_drawdown = (macd_returns + 1).cumprod().div((macd_returns + 1).cumprod().cummax()).sub(1).min()

    performance = performance.append({'Indicator': 'MACD', 'Sharpe Ratio': macd_sharpe_ratio,
                                      'Total Return': macd_total_return, 'Annualized Return': macd_annualized_return,
                                      'Max Drawdown': macd_max_drawdown}, ignore_index=True)

# Sort the performance DataFrame by Sharpe Ratio
performance.sort_values('Sharpe Ratio', ascending=False, inplace=True)

# Save the performance results to a CSV file
performance.to_csv(os.path.join(output_folder, 'indicator_performance.csv'), index=False)

# Create a plot of the total returns for each indicator
plt.figure(figsize=(10, 6))
plt.bar(performance['Indicator'], performance['Total Return'])
plt.xticks(rotation=45, ha='right')
plt.xlabel('Indicator')
plt.ylabel('Total Return')
plt.title('Total Returns by Indicator')
plt.tight_layout()
plt.savefig(os.path.join(output_folder, 'total_returns.png'))
```

OBRÁZEK 31 STATISTICKÁ ANALÝZA #5

```
    # Save the performance results to a CSV file
    performance.to_csv(os.path.join(output_folder, 'indicator_performance.csv'), index=False)

    # Create a plot of the total returns for each indicator
    plt.figure(figsize=(10, 6))
    plt.bar(performance['Indicator'], performance['Total Return'])
    plt.xticks(rotation=45, ha='right')
    plt.xlabel('Indicator')
    plt.ylabel('Total Return')
    plt.title('Total Returns by Indicator')
    plt.tight_layout()
    plt.savefig(os.path.join(output_folder, 'total_returns.png'))

    # Create a plot of the Sharpe Ratio for each indicator
    plt.figure(figsize=(10, 6))
    plt.bar(performance['Indicator'], performance['Sharpe Ratio'])
    plt.xticks(rotation=45, ha='right')
    plt.xlabel('Indicator')
    plt.ylabel('Sharpe Ratio')
    plt.title('Sharpe Ratio by Indicator')
    plt.tight_layout()
    plt.savefig(os.path.join(output_folder, 'sharpe_ratio.png'))

input_folder = "stockDataIndiSmall"
output_folder = "stockResults2"

analyze_stocks(input_folder, output_folder)
```

OBRÁZEK 32 - STATISTICKÁ ANALÝZA #6

```python
import csv
import os
import logging
import yfinance as yf
from PyQt5.QtWidgets import (QApplication, QWidget, QVBoxLayout, QTableWidget, QTableWidgetItem, QPushButton,
                             QTextEdit, QHeaderView, QMainWindow, QLabel)
from PyQt5.QtCore import Qt, QCoreApplication, QEvent, QTimer
from PyQt5.QtGui import QColor
import pickle

def save_selected_stocks(stocks):
    print(stocks)  # add this line to check the contents of stocks
    with open('selected_stocks.pkl', 'wb') as f:
        pickle.dump(stocks, f)

class StockList:
    def __init__(self):
        self.selected_stocks = []

    def add_stock(self, ticker):
        if ticker not in self.selected_stocks:
            self.selected_stocks.append(ticker)

    def remove_stock(self, ticker):
        if ticker in self.selected_stocks:
            self.selected_stocks.remove(ticker)

    def get_stocks(self):
        return self.selected_stocks


class ButtonTableWidgetItem(QWidget):
    def __init__(self, selected_stocks, ticker, parent=None):
        super(ButtonTableWidgetItem, self).__init__(parent)
        self.selected_stocks = selected_stocks
        self.ticker = ticker

        self.button = QPushButton("Add", self)
        self.button.clicked.connect(self.toggle_button)

    def toggle_button(self):
        if self.button.text() == "Add":
            self.button.setText("Remove")
            self.selected_stocks.add_stock(self.ticker)
        else:
            self.button.setText("Add")
            self.selected_stocks.remove_stock(self.ticker)
        save_selected_stocks(self.selected_stocks.get_stocks())
        print(self.selected_stocks.get_stocks())
```

OBRÁZEK 33 - SCREENER #1

```
class ButtonTableWidgetItem(QWidget):
    def __init__(self, selected_stocks, ticker, parent=None):
        super(ButtonTableWidgetItem, self).__init__(parent)
        self.selected_stocks = selected_stocks
        self.ticker = ticker

        self.button = QPushButton("Add", self)
        self.button.clicked.connect(self.toggle_button)

    def toggle_button(self):
        if self.button.text() == "Add":
            self.button.setText("Remove")
            self.selected_stocks.add_stock(self.ticker)
        else:
            self.button.setText("Add")
            self.selected_stocks.remove_stock(self.ticker)
        save_selected_stocks(self.selected_stocks.get_stocks())
        print(self.selected_stocks.get_stocks())

def load_selected_stocks():
    try:
        with open('selected_stocks.pkl', 'rb') as f:
            stocks = pickle.load(f)
    except FileNotFoundError:
        stocks = []
    return stocks


class NumericTableWidgetItem(QTableWidgetItem):
    def __init__(self, value):
        super().__init__(f"{value:.2f}")
        self.value = value

    def __lt__(self, other):
        return self.value < other.value


class LogWidget(QTextEdit):
    def __init__(self):
        super().__init__()
        self.setReadOnly(True)

    def log(self, message, color=None):
        if color:
            self.setTextColor(color)
        self.append(message)
        if color:
```

OBRÁZEK 34 - SCREENER #2

```
class NumericTableWidgetItem(QTableWidgetItem):
    def __init__(self, value):
        super().__init__(f"{value:.2f}")
        self.value = value

    def __lt__(self, other):
        return self.value < other.value


class LogWidget(QTextEdit):
    def __init__(self):
        super().__init__()
        self.setReadOnly(True)

    def log(self, message, color=None):
        if color:
            self.setTextColor(color)
        self.append(message)
        if color:
            self.setTextColor(Qt.black)


class LogHandler(logging.Handler):
    def __init__(self, log_widget):
        super().__init__()
        self.log_widget = log_widget

    def emit(self, record):
        msg = self.format(record)
        color = None
        if record.levelno == logging.ERROR:
            color = QColor(255, 0, 0)
        elif record.levelno == logging.INFO:
            color = QColor(0, 128, 0)
        QCoreApplication.instance().postEvent(self.log_widget, LogEvent(msg, color))


class LogEvent(QEvent):
    EVENT_TYPE = QEvent.Type(QEvent.registerEventType())

    def __init__(self, message, color=None):
        super().__init__(LogEvent.EVENT_TYPE)
        self.message = message
        self.color = color
```

OBRÁZEK 35 - SCREENER #3

```
class LoggerWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.log_widget = LogWidget()
        self.setCentralWidget(self.log_widget)
        self.setWindowTitle('Log Window')
        self.setGeometry(500, 100, 400, 300)

        # Install the custom event filter to handle LogEvent
        self.log_widget.installEventFilter(self)

    def eventFilter(self, source, event):
        if event.type() == LogEvent.EVENT_TYPE:
            self.log_widget.log(event.message, event.color)
            return True
        return super().eventFilter(source, event)

    def setup_logger(self):
        logger = logging.getLogger()
        logger.setLevel(logging.INFO)
        handler = LogHandler(self.log_widget)
        handler.setFormatter(logging.Formatter('%(asctime)s - %(levelname)s - %(message)s'))
        logger.addHandler(handler)
        return logger


def read_tickers_from_csv(files):
    tickers = []
    for file in files:
        file_path = os.path.join('Kernel', 'Stocks', file)
        with open(file_path, newline='') as csvfile:
            reader = csv.reader(csvfile)
            for row in reader:
                tickers.append(row[0])
    return tickers
```

OBRÁZEK 36 - SCREENER #4

```
class StockData(QWidget):
    def __init__(self, tickers, selected_stocks):
        super().__init__()
        self.tickers = tickers
        self.selected_stocks = selected_stocks

    def toggle_stock(self, ticker):
        if ticker in self.selected_stocks:
            self.selected_stocks.remove(ticker)
        else:
            self.selected_stocks.append(ticker)

    def initUI(self):
        layout = QVBoxLayout()
        self.table = QTableWidget()
        layout.addWidget(self.table)
        self.setLayout(layout)
        self.setGeometry(100, 100, 1100, 600)
        self.setWindowTitle('Stock Data')

        refresh_button = QPushButton('Refresh Data', self)
        layout.addWidget(refresh_button)
        refresh_button.clicked.connect(self.load_stock_data)

        self.load_stock_data()

        self.show()

    def load_stock_data(self):
        logger = logging.getLogger()
        logger.info("Loading stock data...")
        data = yf.download(self.tickers, period="1d", group_by="ticker")

        self.table.setRowCount(len(self.tickers))
        self.table.setColumnCount(8)  # Set the column count to 8
        self.table.setHorizontalHeaderLabels(["Ticker", "Open", "High", "Low", "Close", "Volume", "Volume*Price", "Action"])

        timer = QTimer()
        timer.timeout.connect(QCoreApplication.processEvents)
        timer.start(0)
```

OBRÁZEK 37 - SCREENER #5

```
for i, ticker in enumerate(self.tickers):
    try:
        stock_data = data[ticker].iloc[-1]
        self.table.setItem(i, 0, QTableWidgetItem(ticker))

        item = NumericTableWidgetItem(stock_data['Open'])
        self.table.setItem(i, 1, item)

        item = NumericTableWidgetItem(stock_data['High'])
        self.table.setItem(i, 2, item)

        item = NumericTableWidgetItem(stock_data['Low'])
        self.table.setItem(i, 3, item)

        item = NumericTableWidgetItem(stock_data['Close'])
        self.table.setItem(i, 4, item)

        item = NumericTableWidgetItem(stock_data['Volume'])
        self.table.setItem(i, 5, item)

        item = NumericTableWidgetItem(stock_data['Volume'] * stock_data['Close'])
        self.table.setItem(i, 6, item)

        button_item = ButtonTableWidgetItem(self.selected_stocks, ticker)
        self.table.setCellWidget(i, 7, button_item)

        logger.info(f"Ticker {ticker} has been successfully loaded.")
    except KeyError:
        logger.error(f"Error fetching data for {ticker}")

    logger.info(f"Completed row {i+1} of {len(self.tickers)}.")

logger.info("Finished loading stock data.")
self.refresh_data()  # Call refresh_data to resize columns and apply resize mode
```

OBRÁZEK 38 - SCREENER #6

```
    def refresh_data(self):
        self.table.resizeColumnsToContents()
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Interactive)
        for i in range(self.table.columnCount()):
            self.table.horizontalHeader().setSectionResizeMode(i, QHeaderView.Stretch)
        self.table.setSortingEnabled(True)
        self.table.sortItems(6, Qt.DescendingOrder)
        for i in range(10):
            ticker_item = self.table.item(i, 0)
            ticker = ticker_item.text()
            self.selected_stocks.add_stock(ticker)
            button_item = self.table.cellWidget(i, 7)
            button_item.toggle_button()



def main():
    app = QApplication([])
    logger_window = LoggerWindow()
    logger = logger_window.setup_logger()
    logger_window.show()

    # Load selected stocks from file
    selected_stocks = load_selected_stocks()

    stock_list = StockList()
    for ticker in selected_stocks:
        stock_list.add_stock(ticker)

    tickers = read_tickers_from_csv(['AMEX.csv'])
   # tickers = read_tickers_from_csv(['AMEX.csv','NASDAQ.csv','NYSE.csv'])
    stock_data = StockData(tickers, stock_list)
    stock_data.initUI()
    app.exec_()
    save_selected_stocks(stock_list.selected_stocks)  # Save the selected stocks to a file
    return stock_list

if __name__ == '__main__':
    selected_stocks = main()
```

OBRÁZEK 39 - SCREENER #7