Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Automatizované obchodovaní na trhu s kryptoměnami

Bc. Marcel Václav

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Marcel Václav, DiS.

Systémové inženýrství a informatika
Informatika

Název práce

Automatizované obchodovaní na trhu s kryptoměnami

Název anglicky

Automated trading on cryptocurrencies market

Cíle práce

Práce se zabývá problematikou automatického obchodování s kryptoměnami a aktuálním vývojem v oblastí Kryptoměn a blockchainu. Práce je rozdělena na tři hlavní části:

Hlavním cílem práce je vytvoření aplikace, která bude umožňovat uživateli obchodovat na „Binance“ trhu z kryptoměnami a nastavovat automatické obchodování při určitém nastavení parametru nákupu a prodeje.

Metodika

Teoretická část: úvod do problematiky kryptoměn, historie a vývoj. Blockchain, Smart Contracts. Krypto trh a obchodování.

Analytická část: vývoj a předpovídání grafu kryptoměny. Faktory ovlivňující vývoj měny a AI.

Praktická část: Návrh řešení problematiky automatického obchodování na základě analýzy. Vytvoření a popis aplikace na obchodování na Binance.

Doporučený rozsah práce

60-80 stran

Klíčová slova

Bitcoin, UML, Blockchain, API, ARIMA, Strojové učení (AI), Hard Fork, Soft Fork, Scikit

---

Doporučené zdroje informací

A Quick Guide to Cryptocurrency Terms. Cryptocurrencyfacts [online]. 2017 [cit. 2019-04-14]. Dostupné z: https://cryptocurrencyfacts.com/a-quick-guide-tocryptocurrency-terms/

DOSTÁL, P. Pokročilé metody rozhodování v podnikatelství a veřejné správě. Brno: CERM, 2012. 718 s. ISBN 978-80-7204-798-7

Mougayar, William. The business blockchain : promise, practice, and application of the next Internet technology. Hoboken, New Jersey: John Wiley & Sons, Inc, 2016. ISBN 978-1-119-30031-1

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 9. 3. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 13. 3. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 27. 11. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Automatizované obchodovaní na trhu s kryptoměnami" 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 uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne datum odevzdání _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) mému vedoucímu práce za trpělivost a odborný dozor při vedení práce panu Ing. Jiřímu Brožkovi, Ph.D. A dále pak své rodině, zejména svojí Mamině, která mě neustále vedla k dokončení studia. Také kolegům v práci, kterými vycházeli v ústrety.

Automatizované obchodovaní na trhu s kryptoměnami

Abstrakt

Cílem práce bylo zaměřit se na současný trendy v oblastí obchodovaní s kryptoměnami na trhu. Kdy jsme analyzovali problematiku kryptoměn a blockchainu a možnosti obchodování na trhu s kryptoměnami, kdy hlavní myšlenkou bylo automatizování obchodování. Předpokladem proto bylo analyzování možností investování, sběr a analýza časových řad z investičního portfolia a tvorba vhodného postupu pro predikování stavu a rozhodovací funkce softwaru. Během práce jsme objasnili problematiku kryptoměn, možnosti jejich využití a obchodování. V další části jsme se zabývali problematikou Blockchainu a aktuální trendy v oblasti rozvoje. Zjistil jsme jaké jsou možné investiční strategie při obchodování a možnosti analýzy a predikce časové řady. Analyzovali jsme možnosti strojového učení (AI) pro lepší vyhodnocování budoucího vývoje časové řady. Na základě těchto analýz se podařilo vytvořit vhodné řešení pro automatizaci obchodovaní na krypto trhu. Toto navrhnuté řešení je použitelné pro analýzu a umožní uživateli zobrazit vyhodnocené časové řady kryptoměn z jeho portfolia a nastavit automatizované obchodování.

Klíčová slova: Bitcoin, UML, Blockchain, API, ARIMA, Strojové učení (AI), Hard Fork, Soft Fork, Scikit

Automated trading in the Cryptocurrency Market

Abstract

The objective of the thesis was to focus on current trends in the cryptocurrency trading market. We analyzed the problematic of cryptocurrencies and blockchain and possibilities for trading in the cryptocurrency market, with a primary focus on automating trading. Therefore, the prerequisite was examination of investment possibilities, collection and analysis of time series from investment portfolio and development of a suitable procedure for predicting the state and decision-making function of the software. Throughout the thesis, we clarified the problematic of cryptocurrencies, their potential uses and trading. In the next section, we delved into Blockchain problematic and current development trends. We identified possible investment strategies for trading, as well as options for analyzing and predicting time series. We scrutinized the possibilities of machine learning (AI) for better evaluation of the future development of time series. Based on these analyses, we successfully devised a suitable solution for automating trading on the crypto market. The proposed solution is applicable for analysis and enables a user to display evaluated time series of cryptocurrencies from their portfolios and set up automated trading.

Keywords: Bitcoin, UML, Blockchain, API, ARIMA, Machine lerning (AI), Hard Fork, Soft Fork, Scikit

Obsah

1 Úvod

V posledních letech se stali populární součásti investičního trhu kryptoměny. Jsou součástí téměř každého investičního portfolia i když se považují za poměrně rizikovou investici a velké části je toto tvrzení pravdivé. Právě rychlí vzestup a inherentní volatilita je dělá atraktivní a zároveň kontroverzní složkou. My se budeme v této práci věnovat právě kryptoměnám, jejich historii, současným trendům a technikám obchodování s nimi.

Kryptoměny představují novou éru digitálních financí a přinášejí s sebou nové výzvy a příležitosti. V práci se budeme zabývat nejen analýzou historického vývoje a současného stavu kryptoměnových trhů, ale také praktickými aspekty obchodování. Přes svou narůstající popularitu jsou kryptoměny často vnímány jako vysoce riziková aktiva, což je důsledek jejich vysoké volatility a neustálých tržních fluktuací. Tato problematika je rozdělena na několik části.

V teoretické části se věnujeme základním pojmům a teorii kryptoměn, strojového učení a prediktivních metod. Analytická část se poté zaměřuje na porovnání a analýzu různých přístupů k predikci cen, a to vše s cílem definovat nejlepší možné řešení pro naše účely. Praktická část je věnována návrhu technického řešení a strategii v rámci automatizovaného obchodování na trhu s kryptoměnami. Přičemž klíčovým aspektem je vyhodnocení efektivity a úspěšnosti navržených řešení.

Účelem práce je poskytnout čtenářům ucelený vhled do světa kryptoměn, představit jejich potenciál i rizika a nabídnout pragmatický návod na navigaci v tomto dynamickém a někdy nepředvídatelném prostředí. Skrze teoretické poznatky a praktické aplikace se pokusíme odhalit, jak může automatizované obchodování přispět k bezpečnému začlenění kryptoměn do diverzifikovaného investičního portfolia a jaké role mohou hrát v budoucích finančních systémech.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této diplomové práce je návrh a implementace aplikace umožňující uživateli obchodování na kryptoměnové burze Binance. Aplikace poskytne funkcionalitu pro automatické obchodní strategie, které budou reagovat na dynamicky se měnící tržní podmínky a umožní uživatelům nastavit parametry pro automatický nákup a prodej kryptoměn. Součástí práce bude i vývoj prediktivního modelu pro předpověď cenových trendů, jenž bude sloužit jako nástroj pro podporu rozhodování při obchodování.

2.2 Metodika

Metodika práce je založena na kombinaci teoretického výzkumu a praktického softwarového vývoje. Teoretická část zahrnuje rešerši a analýzu odborné literatury zaměřené na kryptoměny, blockchain-ové technologie a metody strojového učení. V praktické části dojde k návrhu systému na základě poznatků získaných v teoretické části a následné implementaci softwarové aplikace. Během vývoje aplikace bude kladen důraz na agilní přístupy a iterativní vývoj, což umožní pružné reagování na změny a optimalizaci funkcí v průběhu celého vývojového cyklu. Součástí metodiky bude také testování funkčnosti a bezpečnosti aplikace, validace prediktivních modelů a jejich integrace do uživatelského rozhraní. Výsledná aplikace bude otestována v reálném čase a bude k ní připravena uživatelská dokumentace.

3 Teoretická východiska

V první části práce si je potřeba říci více k problematice samotného obchodování na trhu s kryptoměnami. Co jsou to kryptoměny a jaký jejich obecný význam a také jak vlastně funguji v z pohledu transakcí a obchodování. Dále pak jejich dnešní podoba a možné využití v budoucnosti.

3.1 Kryptoměny a Kryptografie

3.1.1 Kryptoměny

kryptoměny jsou digitální měny, které jsou založeny na kryptografických principech, což zajišťuje jejich bezpečnost a decentralizaci. Kryptoměny jsou často spojovány s pojmem blockchain, což je způsob ukládání dat, který umožňuje uživatelům bezpečně a transparentně provádět transakce bez potřeby centrálního úložiště nebo prostředníka. (Nakamoto, 2008)

Vysvětlení pojmu kryptoměna je tedy spojeno s konceptem decentralizace a kryptografie. Kryptoměny jsou digitální měny, které jsou zabezpečeny pomocí kryptografických principů a distribuovaného úložiště (blockchainu). To znamená, že transakce jsou ověřovány a zaznamenávány nezávisle na centrální autoritě, což poskytuje větší bezpečnost a transparentnost.

Pojmy jako Bitcoin, Ethereum, Litecoin, Ripple a další jsou dnes dobře známými kryptoměnami, které byly vytvořeny na základě těchto principů. Každá z nich má své vlastní specifikace a účely, ale všechny mají společné to, že jsou decentralizované a založené na kryptografii.

3.1.2 Historie kryptoměn

Historie vývoje kryptoměn se datuje do roku 2008, kdy neznámý autor, který se identifikoval pouze pod pseudonymem Satoshi Nakamoto, publikoval bílou knihu popisující koncept Bitcoinu. Bitcoin byl první decentralizovanou kryptoměnou, která umožňovala lidem posílat peníze přímo mezi sebou bez prostředníků jako jsou banky a finanční instituce.

V roce 2009 byla spuštěna první síť Bitcoinu a začaly se těžit první Bitcoiny. Postupem času se Bitcoin stal stále populárnější a byly vytvořeny další kryptoměny, jako například Litecoin, Ripple, Ethereum a další.

V roce 2017 se hodnota Bitcoinu dramaticky zvýšila a dosáhla historických maxim, což přilákalo pozornost médií a veřejnosti k této nové formě digitálního bohatství. Od té doby se kryptoměny staly stále více populární a jejich využití se rozšířilo mimo pouhou spekulaci na vývoj cen.
(Narayanan, 2016)

3.1.3 Využití kryptoměn

Aplikace kryptoměn se týká využití kryptoměn v praxi. Jedním z hlavních důvodů, proč se lidé zajímají o kryptoměny, je touha po nových způsobech použití a možnostech zisku. Zde je několik příkladů, jak se kryptoměny mohou v praxi využít:

Platební systém: Kryptoměny lze použít jako alternativu k tradičním platebním systémům jako jsou kreditní karty nebo PayPal. Využití kryptoměn jako platebního systému má několik výhod, včetně menších poplatků za transakce a větší anonymity při platbách.

Investice: Kryptoměny se staly populární formou investice. Mnoho lidí nakupuje kryptoměny jako dlouhodobou investici a s nadějí, že budou v budoucnosti víc hodnotné.

Smart contracts: Kryptoměny mohou být použity pro tvorbu tzv. "smart contracts" neboli chytrých smluv, které umožňují automatizované a důvěryhodné uzavírání smluvních dohod.

Těžba: Kryptoměny lze těžit pomocí speciálního hardwaru, což je proces, při kterém se získávají nové mince a transakce jsou ověřovány.

Charity: Některé charitativní organizace přijímají kryptoměny jako dar.

Micropayments: Kryptoměny mohou být použity pro mikrotransakce, jako například platby za jednotlivé části hudby nebo článků.

Identifikace: Kryptoměny mohou být použity pro ověřování identity uživatelů, což by mohlo být užitečné v situacích, kdy je třeba ověřit pravost určitého dokumentu.

Příkladem aplikace kryptoměn je například Bitcoin. Bitcoin je nejznámější a nejstarší kryptoměnou, která byla vytvořena v roce 2009. Bitcoin umožňuje převod peněz mezi uživateli po celém světě bez nutnosti použití banky nebo jiné finanční instituce. Bitcoin využívá technologii blockchain, což je decentralizovaná a distribuovaná databáze, která umožňuje záznam transakcí a ověřování jejich pravosti. (Casey, 2018)

3.1.4 Kryptografie

Kryptoměny využívají principy kryptografie, zejména kryptování a hašování, aby zajistily bezpečnou a anonymní výměnu digitálních aktiv. Kryptování se používá k zabezpečení přenosu dat, jako jsou například transakce s kryptoměnami. Kryptování umožňuje tajné zakódování dat pomocí klíče, který je potřeba k dešifrování dat.

V případě kryptoměn se kryptování používá k zabezpečení transakcí, kdy se data (např. výše transakce, adresy účastníků apod.) šifrují pomocí veřejného a soukromého klíče. Každý uživatel má svůj soukromý klíč, který slouží k dešifrování přijatých dat a podepisování vlastních transakcí. Veřejný klíč je sdílen mezi všemi uživateli a slouží k šifrování dat, která mají být odeslána uživateli.

Hašovací funkce se používají k ukládání a ověřování hesel a ověřovacích kódů. Hašování je proces, při kterém se ze zadaného textu vytvoří hašovací kód, který slouží jako jeho jednoznačná reprezentace. Hašovací funkce jsou nevratné a nelze z nich zpětně získat původní text. Kryptoměny používají hašovací funkce k ověřování platnosti transakcí a k vytváření digitálních podpisů.

Mezi nejznámější kryptoměny, které využívají principy kryptografie, patří Bitcoin, Ethereum, Litecoin a Ripple. Tyto kryptoměny využívají různé algoritmy kryptografie a technologie, které zajišťují bezpečnost a anonymitu transakcí. (Ferguson, 2010)

3.1.4.1 Šifrování a Dešifrování

Šifrování a dešifrování jsou klíčové pojmy v oblasti kryptografie a jsou nezbytné pro zabezpečení dat a soukromí. V kontextu kryptoměn se tyto pojmy týkají především zabezpečení transakcí a peněženek.

Šifrování je proces převodu čitelného textu, zvaného otevřený text, na nečitelný tvar, zvaný šifrový text. Cílem šifrování je zajistit, aby pouze osoby s příslušným klíčem mohly dešifrovat šifrovanou zprávu a získat tak původní otevřený text. Existuje mnoho druhů šifrovacích algoritmů, jako například Caesarova šifra, substituční šifry nebo moderní symetrické šifry jako AES (Advanced Encryption Standard) nebo Blowfish.

Dešifrování je opačný proces, který převádí šifrový text zpět na otevřený text. Proces dešifrování vyžaduje znalost správného klíče, kterým byl text šifrován.

V kontextu kryptoměn se používají asymetrické šifry, které vyžadují dva klíče: veřejný a soukromý. Veřejný klíč je k dispozici každému, kdo chce zasílat kryptoměnu, a

slouží k šifrování zprávy. Soukromý klíč, který je znám pouze majiteli peněženky, slouží k dešifrování zprávy a potvrzení platby.

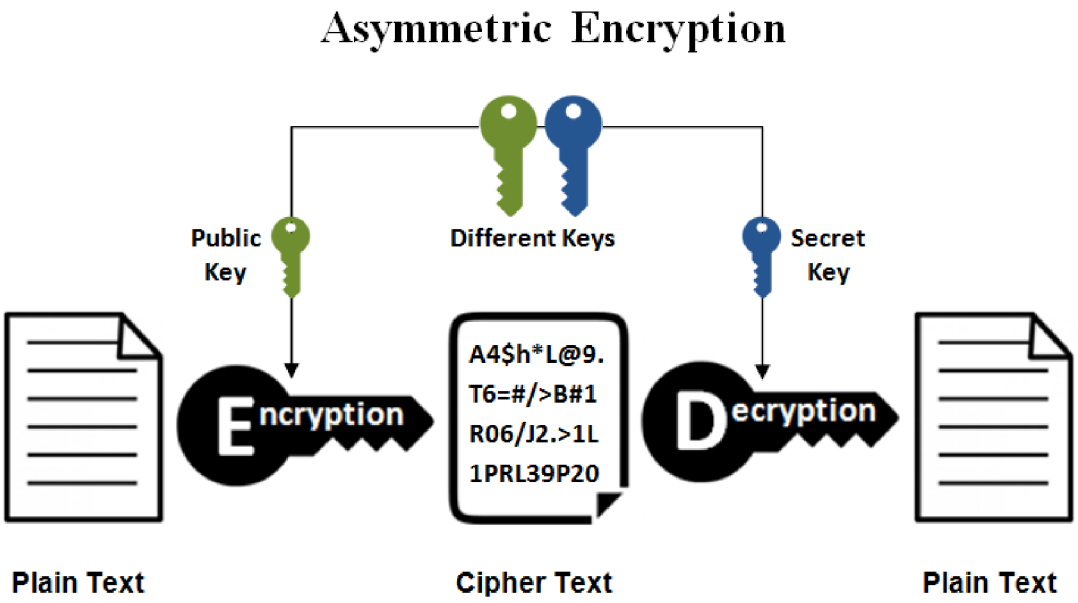
Hašovací funkce jsou dalším důležitým prvkem kryptografie a kryptoměn. Tyto funkce přijímají vstupní data libovolné délky a vrací pevně danou délku výstupu, který se nazývá hash. Hašovací funkce jsou používány pro zabezpečení kryptoměn tím, že hašují transakce a uložené údaje o peněženkách, což zajišťuje, že tyto údaje nelze měnit ani padělat. Populární hašovací funkce pro kryptoměny je SHA-256 (Secure Hash Algorithm). (Stallings, 2013)

3.1.4.2 Soukromý a veřejný klíč

V kryptografii se používají dva typy klíčů pro šifrování a dešifrování zpráv – veřejný klíč a soukromý klíč. Tyto klíče jsou navrženy tak, aby bylo snadné zasílat zprávy mezi lidmi bez nutnosti sdílet stejný tajný klíč.

Veřejný klíč, také známý jako šifrovací klíč, je klíč, který je sdílen s ostatními. Tento klíč může být použit k zašifrování zprávy, která je určena určitému příjemci. Když je zpráva zašifrována veřejným klíčem, může ji dešifrovat pouze ten, kdo má k dispozici soukromý klíč, který odpovídá veřejnému klíči. Veřejný klíč je tedy veřejně dostupný a může být sdílen s kýmkoli.

Soukromý klíč, také známý jako dešifrovací klíč, je klíč, který je držen tajně a slouží k dešifrování zpráv, které byly zašifrovány pomocí odpovídajícího veřejného klíče. Tento klíč je vlastněn pouze příjemcem zprávy a nikdo jiný by neměl mít přístup k této informaci. (Stallings, 2013)



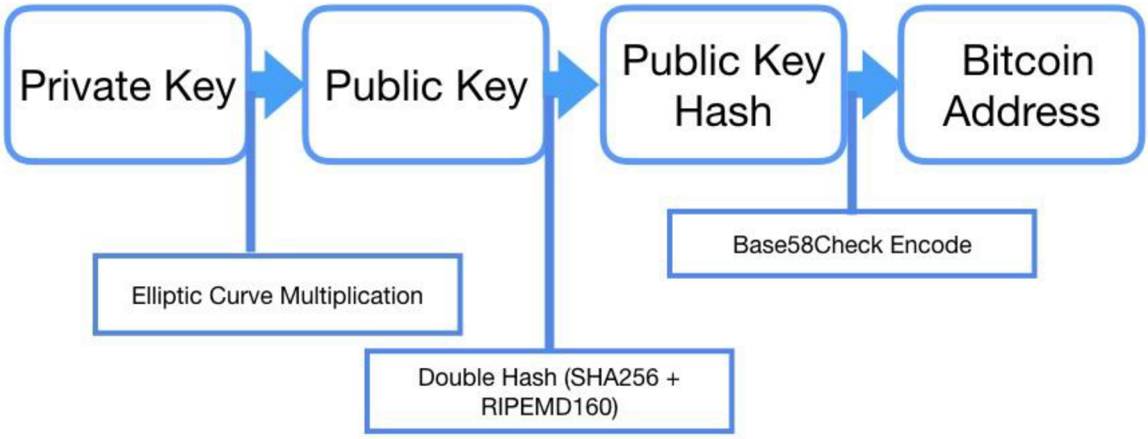
Obrázek 1 – využití primárního a veřejného klíče pro šifrování/dešifrování

Ilustrační obrázek níže ukazuje, jak může být veřejný a soukromý klíč použit pro bezpečné šifrování a dešifrování zpráv:

Při použití veřejného klíče k zašifrování zprávy je tato zpráva šifrována pomocí veřejného klíče a poté odeslána příjemci zprávy. Příjemce zprávy poté použije svůj soukromý klíč k dešifrování zprávy. Tímto způsobem může být zajištěna bezpečnost při přenosu zpráv, aniž by bylo nutné sdílet tajné klíče mezi odesílatelem a příjemcem.

Použití veřejného a soukromého klíče se používá v kryptoměnách k podepisování a ověřování transakcí. Každá kryptoměna má svůj vlastní algoritmus pro šifrování a dešifrování, který využívá princip veřejného a soukromého klíče.

Když uživatel chce poslat kryptoměnu jiné osobě, tak musí podepsat transakci svým soukromým klíčem, který slouží k ověření identity uživatele a podepsání transakce. Následně se použije veřejný klíč příjemce, aby se transakce mohla úspěšně realizovat. Tento proces se nazývá asymetrická kryptografie a je základem pro zabezpečení transakcí v kryptoměnové síti. (Jonathan Katz, 2014)



Obrázek 2 - asymetrické šifrování v blockchainu

Zde je ilustrační obrázek, který ukazuje, jak funguje proces asymetrické kryptografie v kryptoměnové síti:

3.1.4.3 Hashovací funkce

Hashovací funkce převádějí zprávu (textový řetězec) libovolné délky na pevnější řetězec o pevné délce. Výsledkem hashovací funkce je zpravidla jednosměrná funkce, což znamená, že nelze zpětně získat původní zprávu z výstupu hashovací funkce. (Nakamoto, 2008)

V kryptoměnách jsou hashovací funkce využity pro ověřování správnosti transakcí a zabezpečení celého blockchainu. Každá transakce je hashovaná a výsledný hash se pak používá jako unikátní identifikátor této transakce. Poté jsou jednotlivé transakce seskupeny do bloků, kde každý blok obsahuje hash předchozího bloku. To zajišťuje, že bloky jsou pevně spojené v řetězci a každá změna v jednom bloku by měla být odhalena jako neplatná, protože by změnila hash předchozího bloku a tím by zničila celou strukturu blockchainu. (Narayanan, 2016 stránky 30-40)

3.1.5 Blockchain

Samotný blockchain je často spojován s kryptoměnami a je považován za jednu z hlavních inovací v oblasti finančních technologií. Blockchain umožňuje ukládat data do sítě nezávislé na jednom centrálním místě, což zvyšuje bezpečnost a decentralizaci. Díky tomu může být blockchain využíván k vytváření a správě kryptoměn, ale také k dalším účelům, jako je například ukládání dat o vlastnictví nebo ověřování identit. V této kapitole

se budeme dále zabývat principy blockchainu, jeho aplikacemi a výzvami, které s ním souvisejí.

Blockchain umožňuje transparentní a bezpečný přenos digitálních aktiv, jako jsou například kryptoměny, bez nutnosti důvěry v jednu centrální autoritu. To znamená, že uživatelé si mohou přímo převádět své digitální aktiva bez prostředníka a věřit, že informace jsou uloženy bezpečně a nelze je libovolně změnit.

V souvislosti s kryptoměnami je blockchain klíčovou technologií, která umožňuje zabezpečit transakce a zajišťuje, že kryptoměny nelze libovolně kopírovat či padělat. Blockchain však najde uplatnění i mimo oblast kryptoměn a je používán například v logistice, zdravotnictví nebo ve volbách.

Princip fungování blockchainu spočívá v decentralizovaném ukládání dat. To znamená, že žádná centrální autorita nemá kontrolu nad daty a žádný účastník sítě nemá monopol na přístup k datům. Namísto toho jsou data uložena v distribuované síti uzlů (nodes) a každý uzel má kopii celého blockchainu.

Každá transakce, která se provede na blockchainu, je ověřena sítí uzlů a přidána do bloku, který je následně připojen ke stávajícímu blockchainu. Každý blok obsahuje hash předchozího bloku, což zajišťuje, že všechny bloky jsou propojeny a nelze je změnit nebo vymazat bez toho, aby byl narušen celý řetězec. Tento mechanismus zajišťuje bezpečnost a integrity dat na blockchainu.

Dalším principem blockchainu je konsensus. To znamená, že uzly v síti musí dosáhnout shody ohledně toho, která transakce je platná a která není. Existují různé algoritmy konsensu, například **proof-of-work**, **proof-of-stake** nebo **delegated proof-of-stake**, které určují, jakým způsobem se sítě dosahuje konsensu.

V neposlední řadě, blockchain umožňuje anonymitu a pseudonymitu transakcí. To znamená, že účastníci mohou provádět transakce pod pseudonymy, které jsou spojeny pouze s jejich veřejnými klíči. To poskytuje vysokou míru ochrany soukromí a zároveň zabraňuje podvodům a zneužití. (Tapscott, 2016)

3.1.5.1 Využití Blockchainu

Kryptoměny a digitální platby: nejznámějším využitím blockchainu jsou kryptoměny jako Bitcoin, Ethereum a další, které umožňují rychlé, levné a bezpečné digitální transakce bez účasti tradičních finančních institucí. Podle statistik CoinMarketCap měla kryptoměna

Bitcoin v lednu 2022 kapitalizaci trhu přes 700 miliard dolarů a Ethereum přes 400 miliard dolarů. Denní počet transakcí Bitcoinu se v lednu 2022 pohyboval kolem 300 000.

Smart kontrakty: blockchain umožňuje vytváření tzv. smart kontraktů, což jsou digitální smlouvy, které se automaticky provádějí, když jsou splněny určité podmínky. Tyto smlouvy mohou být využity v mnoha oblastech, například v logistice, pojišťovnictví, realitách, vývoji softwaru a mnoha dalších.

Správa identit: blockchain může být použit pro správu digitálních identit. Díky blockchainu mohou uživatelé mít kontrolu nad svými osobními daty a poskytovat je jen těm, kteří mají oprávnění. To může být užitečné v oblasti zdravotnictví, bankovnictví, vládních institucí a dalších.

Crowdfunding: Blockchain umožňuje snadnější a levnější crowdfunding, tedy financování projektů získáváním menších částek od většího počtu lidí. Využívá se k tomu tzv. Initial Coin Offerings (ICO), kdy nová kryptoměna vydává své mince nebo tokeny, které mohou být zakoupeny investory. Tyto mince nebo tokeny poté slouží jako platidlo uvnitř projektu. Mezi příklady kryptoměn, které byly vydány pomocí ICO, patří například Ethereum, EOS nebo Tezos.

Logistické a zabezpečovací řešení: Blockchain lze využít pro sledování logistiky a zabezpečení dodávek. Například platforma VeChain využívá blockchain k zaznamenávání informací o původu a průběhu dopravy produktů, což umožňuje větší transparentnost a snadnější identifikaci problémů.

Kapitalizace trhu a počet transakcí pro jednotlivé kryptoměny se neustále mění a může být různý v závislosti na aktuální situaci na trhu. Pro aktuální informace o kapitalizaci trhu a počtu transakcí lze využít specializované webové stránky a platformy, jako například CoinMarketCap nebo CoinGecko. Tyto platformy poskytují také další statistiky a informace o jednotlivých kryptoměnách. (Narayanan, 2016)

3.1.5.2 Hard Fork a Soft Fork

Hard Fork a Soft Fork jsou pojmy, které se používají v souvislosti s blockchainem a změnami, které mohou být v něm provedeny. Obvykle se provádějí k řešení nějakého problému v síti blockchainu, například zvýšení kapacity transakcí, změnu pravidel pro těžení nebo zlepšení ochrany sítě. (Narayanan, 2016 stránky 110-120)

Hard Fork znamená, že se kryptoměna rozdělí na dvě větve, přičemž se oba vývoje od sebe oddělí natolik, že již nemohou sdílet stejný blockchain. To znamená, že každý

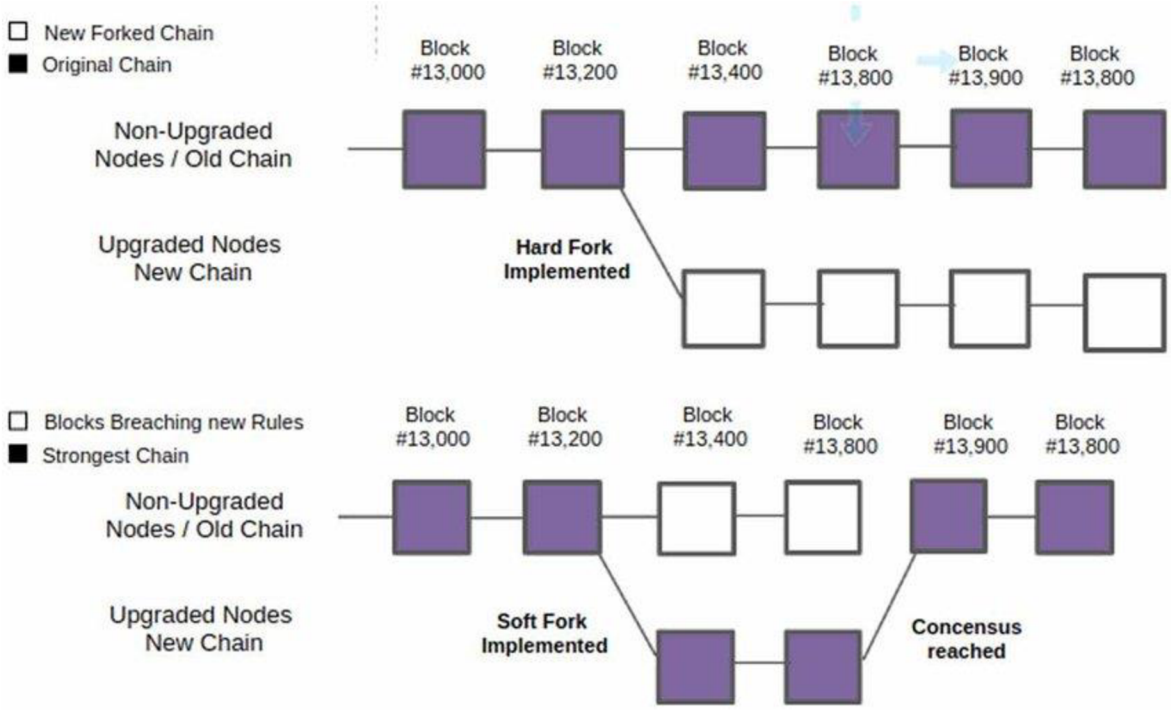
uživatel, který vlastnil tokeny původní kryptoměny, bude mít stejný počet tokenů na obou nových blockchainech. Příkladem Hard Forku je například Bitcoin Cash (BCH), který se oddělil od Bitcoinu (BTC) v srpnu 2017.

Soft Fork znamená, že se kryptoměna upraví, ale blockchain se nadále sdílí. To znamená, že každý uživatel, který vlastní tokeny této kryptoměny, bude mít nadále stejný počet tokenů na stejném blockchainu. Příkladem Soft Forku je například SegWit (Segregated Witness), který byl implementován do Bitcoinu v roce 2017. (Antonopoulos, 2014)

Hard Fork a Soft Fork mohou mít různé důsledky na trh kryptoměn, v závislosti na konkrétní situaci. Pokud se Hard Fork děje v důsledku neshod v komunitě ohledně směru vývoje kryptoměny, může to vést k vytvoření dvou konkurenčních kryptoměn a rozdělení komunity. To může mít vliv na hodnotu původní kryptoměny a na celkovou hodnotu kryptotrhu.

Soft Fork na druhé straně může být pouze menší změnou, která nemá příliš velký vliv na trh. V každém případě se však doporučuje, aby uživatelé držící kryptoměnu během Hard Forku drželi své tokeny na adresách, kde mají kontrolu nad soukromými klíči, aby se předešlo ztrátě tokenů při rozdělení blockchainu.

Statistiky ukazují, že od počátku existence kryptoměn se stalo více než 1000 Hard Forků. Největší a nejznámější Hard Forky jsou Bitcoin Cash (BCH) a Ethereum Classic (ETC). Mezi největší Soft Forky patří zmíněný SegWit pro Bitcoin a také Byzantium pro Ethereum.



Obrázek 3 - koncept Hard Fork a Soft Fork

Tento obrázek ukazuje, že Hard Fork vytváří novou větev blockchainu, která není kompatibilní s původní větví, zatímco Soft Fork využívá existující větev blockchainu a změny jsou zpětně kompatibilní s původní větví.

3.1.6 Obchodování s kryptoměnami

Mají opodstatněné využití v těchto odvětvích:

Platby: Jednou z nejvýznamnějších aplikací kryptoměn je v oblasti platby. Kryptoměny umožňují uživatelům posílat a přijímat platby rychleji a levněji než tradiční bankovní převody. Například, Bitcoin umožňuje převody mezi lidmi po celém světě bez nutnosti použití banky nebo jiných finančních institucí.

Kryptoměny také umožňují lidem financovat projekty přes crowdfundingové platformy, jako je Kickstarter nebo Indiegogo. Kryptoměny usnadňují přeshraniční transakce a umožňují projektům získat finanční podporu od lidí po celém světě.

Kryptoměny umožňují uživatelům obchodovat s různými aktivy, jako jsou akcie, zlato, státní dluhopisy atd. na decentralizovaných burzách. Ethereum například umožňuje tvorbu chytrých smluv a tokenů, které mohou být použity pro různé účely.

Elektronický obchod: Kryptoměny také umožňují platby v elektronickém obchodě a nabízejí výhody, jako je rychlost, snadnost a nízké náklady. Například, některé obchody přijímají Bitcoin nebo další kryptoměny jako platbu za zboží a služby.

Finanční služby: Kryptoměny mohou také poskytnout nové finanční služby, jako jsou půjčky a úvěry, které nejsou dostupné v tradičním finančním systému. Například, některé platformy nabízejí půjčky v kryptoměnách za nižší úrokové sazby než tradiční banky.

Hodnota trhu s kryptoměnami se v posledních letech rychle zvyšuje. K 31. lednu 2022 byla celková tržní kapitalizace kryptoměn více než 1,5 bilionu dolarů. Největší kryptoměnou podle tržní kapitalizace je Bitcoin s hodnotou přes 900 miliard dolarů, následovaný Ethereum (Buterin, 2014), Binance Coin, Solana a Cardano.

- Bitcoin je založen na technologii blockchain a využívá kryptografii pro zabezpečení transakcí. Je považován za první a nejznámější kryptoměnu. Bitcoin umožňuje rychlé a anonymní převody bez účasti bankovních institucí. Jeho kryptografické algoritmy a decentralizovaný charakter zajišťují bezpečnost a transparentnost transakcí. Nicméně, v posledních letech se objevily obavy ohledně jeho ekologické zátěže a rychlosti transakcí. (Antonopoulos, 2014)

- Ethereum je platforma pro vývoj decentralizovaných aplikací (DApps), která umožňuje programátorům psát chytré smlouvy a spouštět je na blockchainu. Ethereum má svou vlastní kryptoměnu Ether, která se používá pro placení poplatků za transakce v síti. Ethereum nabízí mnoho výhod, jako jsou chytré smlouvy, rychlé a levné transakce a široká škála použití.

- Cardano je další kryptoměna, která využívá blockchain. Založena byla roku 2017 a je často považována za konkurenta Etherea. Cardano se pyšní tím, že je jednou z nejrychlejších a nejbezpečnějších kryptoměn na trhu. Nabízí také

škálu vývojářských nástrojů a umožňuje tvorbu chytrých smluv a decentralizovaných aplikací.

- Solana je relativně nová kryptoměna, která se objevila až v roce 2020. Jedná se o rychlou a škálovatelnou kryptoměnu, která umožňuje transakce v řádu tisíců za vteřinu. Solana používá technologii proof of history (důkaz historie), která umožňuje rychlou validaci transakcí a zpracování velkého množství dat.
- Binance Coin je kryptoměna vytvořená společností Binance, která je jednou z největších kryptoměnových burz na světě. Binance Coin se používá pro platby na této burze a je také využívána pro zaplacení poplatků. Kromě toho má Binance Coin i další využití, jako například pro financování projektů, které jsou spuštěny na platformě Binance Launchpad.

Každá z kryptoměn má své specifické vlastnosti, ale všechny jsou postaveny na technologii blockchain. Bitcoin je považován za digitální ekvivalent zlata a je navržen tak, aby byl řízen decentralizovaně, tedy bez centrálního řídícího orgánu. Ethereum, na druhé straně, je platformou pro vývoj decentralizovaných aplikací a umožňuje programátorům vytvářet tzv. smart kontrakty. Cardano se zaměřuje na škálovatelnost a rychlost transakcí, zatímco Solana klade důraz na vysokou rychlost a nízké poplatky za transakce. (Tapscott, 2016)

3.1.7 Kryptoměnové burzy

Burza je platforma, která umožňuje nákup a prodej kryptoměn. Obvykle funguje jako tržiště, kde se setkávají kupující a prodávající a uzavírají obchody za určitou cenu. Kryptoburzy mohou být centralizované nebo decentralizované a mohou nabízet různé funkce a nástroje pro obchodování. Existuje mnoho kryptoburz po celém světě. Některé z největších a nejznámějších burz zahrnují:

- Binance: Jedna z největších krypto-burz na světě. Nabízí obchodování s velkým množstvím kryptoměn a také vlastní kryptoměnu Binance Coin (BNB).
- Coinbase: Jedna z nejpopulárnějších burz pro nákup a prodej kryptoměn v USA. Nabízí také krypto-peněženku a další služby.

- Kraken: Mezinárodní burza se sídlem v USA. Nabízí obchodování s mnoha kryptoměnami a také možnost obchodování s pákou.
- Bitfinex: Burza se sídlem v Hongkongu a obchodující s mnoha kryptoměnami, včetně Tether (USDT), který je vázán na americký dolar.
- Huobi: Burza se sídlem v Singapuru. Nabízí obchodování s mnoha kryptoměnami a také vlastní kryptoměnu Huobi Token (HT).

Hlavní funkcí burzy je umožnit kupujícím a prodávajícím kryptoměn uzavírat obchody a vyměňovat kryptoměny za **fiatové peníze** nebo jiné kryptoměny.
Kryptoburzy mohou nabízet různé platební metody pro nákup kryptoměn, jako jsou bankovní převody, kreditní karty, PayPal a další.
Některé burzy nabízejí pokročilé nástroje pro obchodování, jako jsou limitní a tržní objednávky, stop-loss a další. Hodně nabízejí také krypto-peněženky pro ukládání kryptoměn. (Narayanan, 2016 stránky 140-267)

3.1.7.1 Bezpečnost trhu

Bezpečnost je jedním z nejvýznamnějších faktorů, které by měly hrát roli při výběru krypto-burzy. Mezi nejčastější bezpečnostní rizika patří krádeže, hacking, phishing, phishing, ztráta kódů a podobně. Některé z nejbezpečnějších krypto-burz na trhu nabízejí různé bezpečnostní funkce, jako například:
Dvou-faktorová autentizace (2FA): Krypto-burzy mohou požadovat dvou-faktorovu autentizaci, která vám umožní přidat další úroveň zabezpečení při přihlašování do svého účtu. Tato funkce obvykle vyžaduje, abyste zadali své uživatelské jméno a heslo a následně potvrdili svůj vstup pomocí ověřovacího kódu zaslaného na váš telefon nebo pomocí jiného ověřovacího zařízení.
Platební autorizace: Krypto-burzy mohou také nabízet platební autorizaci, což znamená, že při každé transakci musíte potvrdit svou identitu a platbu prostřednictvím autorizačního kódu nebo hesla.
Kryptografické úložiště: Některé krypto-burzy ukládají kryptoměny v kryptograficky zabezpečených úložištích, která jsou těžko zranitelná pro útoky hackerů.
Cold-storage: Mnoho krypto-burz nabízí také možnost cold-storage, což je metoda uložení kryptoměn offline, což znamená, že k nim nelze přistupovat prostřednictvím internetu.

Tato metoda zajišťuje, že vaše kryptoměny jsou chráněny před hackery a jinými kybernetickými útoky. (Narayanan, 2016 stránky 120-121)

3.1.7.2 Kryptoměnové peněženky

V oblasti kryptoměnových peněženek existuje mnoho různých typů, od hot peněženek, které jsou připojeny k internetu, až po cold peněženky, které jsou uloženy offline. Zde jsou některé z nejpoužívanějších typů kryptoměnových peněženek:

- Desktopové peněženky: Tyto peněženky jsou instalovány přímo na vašem počítači a umožňují ukládání kryptoměn offline. To znamená, že klíče jsou uchovávány na vašem počítači a nejsou připojeny k internetu, což poskytuje vyšší úroveň bezpečnosti než online peněženky. Nicméně, pokud váš počítač není zabezpečen dostatečně, může být ohrožen malwarem nebo hackingem.
    - Mezi nejpopulárnější desktopové peněženky patří Electrum, Armory a Bitcoin Core pro Bitcoin, Míst pro Ethereum a Litecoin Core pro Litecoin.
- Hardwarové peněženky: Tyto peněženky jsou fyzická zařízení, která umožňují ukládání kryptoměn offline. Klíče jsou uloženy na hardwarovém zařízení, jako je například USB klíč nebo podobné zařízení. Hardwarové peněženky jsou považovány za nejbezpečnější možnost ukládání kryptoměn, protože jsou odpojeny od internetu, což minimalizuje riziko hackingu nebo malware. Některé z nejpopulárnějších hardwarových peněženek jsou Ledger Nano S, Trezor a KeepKey.

Při výběru peněženky je důležité zvážit, jakou úroveň bezpečnosti chcete, jaké kryptoměny chcete ukládat a jak často chcete k nim přistupovat. Pokud plánujete ukládat velké množství kryptoměn, je nejlepší zvážit investici do hardwarové peněženky. (Tapscott, 2016)

3.2 Strojové učení a predikce

V rámci automatizování obchodování je důležité pracovat s vývojem aktuálních a historických událostí a vývoje trendů cenových hladin u konkrétní potenciální

akce/komodity/kryptoměny se kterou se hodláme vykonat obchod(nákup/prodej). V rámci analýzy se zabýváme predikací budoucího vývoje časové řady na základě její trendové funkce. Na predikování funkce (časové řady) se využívají různé prognostické metody a metody predikce o kterých si více řekneme v následujících kapitolách. Na řešení těchto úkolů se často využívá umělá inteligence, která je schopná zpracovat a vyhodnotit dané úkoly a na základě nich se dále učit a zlepšovat svoji dovednost. (Warwick, 2010 pp. 1-20)

3.2.1 Umělá inteligence

je obor informatiky a strojového učení, který se zabývá vývojem počítačových systémů schopných provádět úkoly, které by vyžadovaly lidskou inteligenci, jako například rozpoznávání obrazu, rozpoznávání řeči, plánování, rozhodování, překlad jazyků a mnoho dalších. Umělá inteligence může být dále rozdělena do několika podoborů, jako jsou například strojové učení, přirozené jazykové zpracování (NLP), počítačové vidění a robotika.

Strojové učení (ML) je větví umělé inteligence, která se zabývá návrhem algoritmů, které umožňují počítači "učit se" na základě dat a zkušeností. Toto učení umožňuje stroji rozpoznat vzorce v datech a vytvořit přesné predikce nebo klasifikace. Mezi nejpopulárnější algoritmy strojového učení patří rozhodovací stromy, klasifikační modely, neuronové sítě a hluboké neuronové sítě.

Přirozené jazykové zpracování (NLP) se zabývá zpracováním lidské řeči a psaného textu počítačem. Tento obor se využívá například pro automatický překlad jazyků, analýzu sentimentu v textech nebo pro tvorbu chatbotů.

Počítačové vidění se snaží počítač naučit rozpoznávat a interpretovat obrázky a videa. Tento obor se využívá například pro rozpoznávání tváří, aut, zvířat, či pro analýzu zdravotních snímků.

Robotika je obor umělé inteligence, který se zabývá vývojem robotů a umělých agentů, kteří jsou schopni plánovat a provádět úkoly v reálném světě.

Jeden z příkladů umělé inteligence, který se v poslední době stal velmi populárním, jsou chatboti. Chatboti jsou programy, kteří mohou simulovat lidskou konverzaci a odpovědět na otázky a požadavky uživatelů. Tyto chatboty používají přírodní jazykové zpracování a strojové učení pro zlepšení své schopnosti odpovědět na dotazy a rozumět lidskému jazyku.

Dalším příkladem umělé inteligence jsou autonomní vozidla, která mohou řídit samostatně bez lidského řidiče. Tyto vozidla používají řadu senzorů, jako jsou radar, lidar a kamery, k detekci překážek a k plánování trasy.
Při vývoji umělé inteligence se využívají různé přístupy a techniky, jako jsou strojové učení, neuronové sítě, evoluční algoritmy, fuzzy logika, expertní systémy a další. Výsledky výzkumu a aplikace umělé inteligence mají obrovský potenciál pro rozvoj a inovaci v mnoha oblastech, jako je zdravotnictví, výroba, doprava, obchod, finance a další.

3.2.2 Strojové učení

je podobor umělé inteligence, který se zaměřuje na tvorbu algoritmů a modelů, které dokáží na základě dat naučit se určitým vztahům a pravidelnostem a tyto znalosti pak použít pro predikci budoucích hodnot nebo rozhodování.
Strojové učení lze rozdělit do tří hlavních kategorií:

- Super vizované učení (Supervised Learning)
  V učení s učitelem jsou k dispozici trénovací data, která obsahují označení (label), tj. správné výstupy pro každý vstupní datový bod. Algoritmy strojového učení se učí na těchto datech a jejich cílem je předpovídat výstup pro nová, dosud neviděná data.
- Naučení bez učitele (Unsupervised Learning)
  V učení bez učitele nejsou k dispozici označená trénovací data. Algoritmy se snaží najít strukturu nebo vzorce v datech a často jsou využívány k seskupování (clustering) podobných datových bodů.
- Posílené učení (Reinforcement Learning)
  učení je agent umístěn v prostředí a snaží se naučit určité chování, aby maximalizoval svou odměnu. Algoritmy v tomto případě se učí prostřednictvím pozorování, zkoušení a zpětné vazby na základě získané odměny.

Mezi klíčové aspekty strojového učení patří výběr vhodného modelu, správná úprava dat pro trénování, nalezení vhodných hyper-parametrů a validace výsledného modelu. (François, 2019 stránky 94-115)

3.2.2.1 Supervizované učení

Supervizované učení je metoda strojového učení, kde algoritmus je trénován na základě předem označkovaných (označených) dat. Tyto označkované data zahrnují jak vstupní data (tj. funkce, příznaky nebo znaky), tak i očekávané výstupy (tj. výsledek, kategorie nebo třída). Cílem supervizovaného učení je naučit algoritmus odhadovat výstupy pro nová vstupní data, která nebyla v trénovacích datech.

Příkladem supervizovaného učení může být klasifikace e-mailů jako spam nebo ne-spam. Algoritmus by byl trénován na základě označkovaných dat, která obsahují text e-mailů a jejich označení jako spam nebo ne-spam. Na základě toho by algoritmus mohl odhadnout, zda nový e-mail je spam nebo ne.

Tabulka níže ukazuje jednoduchý příklad trénování algoritmu pro klasifikaci květin. Každá květina je popsána třemi vlastnostmi (délka a šířka okvětních lístků a kališní lístky) a má přiřazenu kategorii (setosa, versicolor nebo virginica). Algoritmus se učí na základě těchto označkovaných dat a může pak odhadovat kategorii nových květin na základě jejich vlastností.

Příkladem může být nasledující confusion matrix(čtvercová matice) pro klasifikaci květin:

Tabulka 1 - čtvercová matice pro klasifikaci květin

	Predikované třídy		
	Setosa	VersiColor	Virginica
Skutečné třídy			
Setosa	20	0	0
VersiColor	0	18	2
Virginica	0	1	19

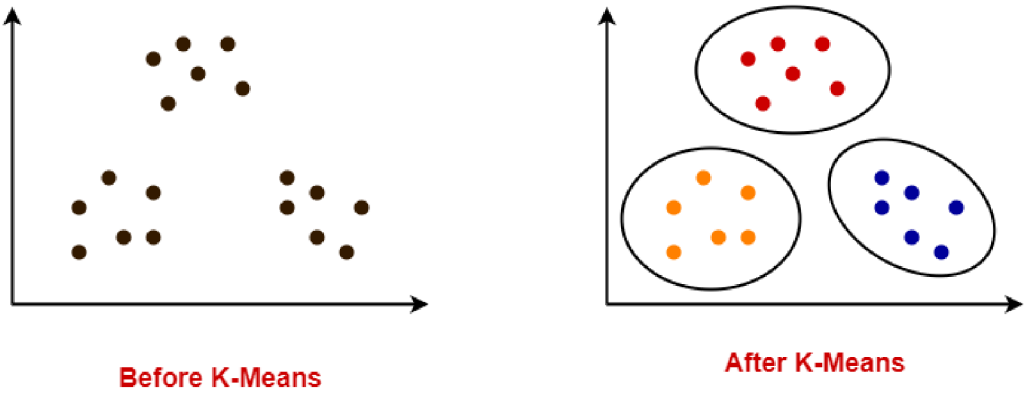
„Matice ukazuje 6e klasifik8tor má 100% přesnost v případě třídy "Setosa", ale není tak přesný u tříd "VersiColor" a "Virginica". Kromě čtvercové matice se používa třeba accuracy, nebo recall a F1-Score. Ty také dokážou zhodnotit kvalitu z různých perspektiv."

3.2.2.2 Naučení bez učitele

Nesupervizované učení je typ strojového učení, při kterém algoritmy hledají struktury a vzorce v datech bez předchozí znalosti výsledků. Tato metoda je často využívána pro objevování skrytých vzorců a struktur v datech a je využívána v mnoha oblastech, jako je například rozpoznávání obrazů, analýza dat, doporučovací systémy a mnoho dalšího.
Mezi příklady nesupervizovaného učení patří například shlukování (clustering), redukce dimenzionality (dimensionality reduction) nebo generativní modelování (generative modeling).
Shlukování je proces, při kterém jsou data rozdělena do skupin nebo shluků podle podobnosti mezi nimi. Tento proces se využívá pro objevování skrytých struktur v datech, jako jsou například skupiny podobných zákazníků nebo obrazů. Mezi populární algoritmy shlukování patří k-means, hierarchické shlukování a DBSCAN. (Křen, 2015 stránky 1-70)

Pro příklad k-means skládá z následujících kroků:

1. Náhodně zvolte k počátečních centroidů (středy) shluků.
2. Každý bod data-setu přiřaďte k nejbližšímu centroidu a vytvořte tak shluky.
3. Spočítejte nové středy shluků jako průměr všech bodů v daném shluku.
4. Pokud se středy shluků již nezměnily (tolerance je splněna) nebo byl dosažen maximální počet iterací, algoritmus skončí. Jinak se vraťte na krok 2.

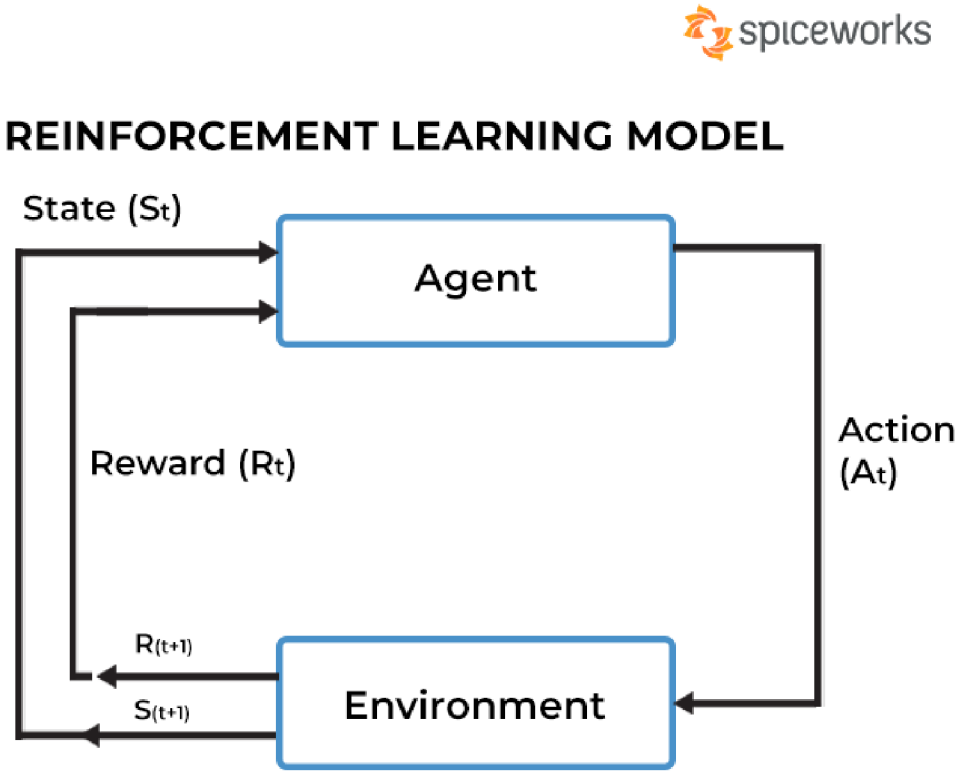


Obrázek 4 - příklad učení bez dohledu

Redukce dimenzionality slouží k redukci počtu proměnných v data-setu. Může být využívána k odstranění šumu, snížení náročnosti výpočtů nebo zobrazení dat v prostoru s menším počtem dimenzí, což umožňuje vizualizaci dat. Mezi populární algoritmy redukce dimenzionality patří PCA, t-SNE a UMAP.
Generativní modelování se používá k vytváření nových dat na základě dat existujících. Tyto modely jsou schopny generovat nová data, která odpovídají rozdělení pravděpodobnosti dat, na kterých byly trénovány. Mezi populární algoritmy generativního modelování patří variational autoencoders a generative adversarial networks (GANs).

3.2.2.3 Posílené učení

je metoda strojového učení, která se snaží naučit strojový systém, jakým způsobem má jednat v daném prostředí, aby maximalizoval získanou odměnu. Tato metoda se používá zejména v oblastech, kde je nemožné získat anotovaná data, nebo kde jsou data velmi drahá a nepraktická k anotaci.
Posílené učení se řídí základním principem trial and error (pokus omyl), kdy strojový systém zkouší různé akce a na základě získané zpětné vazby se snaží naučit, jakým způsobem reagovat na určité situace. Zpětná vazba je zpravidla udělena v podobě odměny, kterou systém dostane za určitou akci v daném prostředí.
Při posíleném učení se pracuje s tzv. agentem, který se nachází v daném prostředí a má za úkol maximalizovat svou odměnu. Agent přijímá vstupní data z prostředí, na základě kterých rozhoduje, jakou akci podnikne, a získá tak určitou odměnu. Cílem agenta je naučit se takovým způsobem jednat v prostředí, aby maximalizoval svou celkovou odměnu.
Příkladem aplikace posíleného učení může být naučení robota hrát hry jako je například šachy, Go nebo StarCraft. Robot postupně zkouší různé tahy a na základě zpětné vazby v podobě výhry nebo prohry se snaží naučit, jakým způsobem hrát lépe. (Křen, 2016 stránky 5-100)



Obrázek 5 - ilustrace posíleného učení – Zdroj: (Richard S. Sutton and Andrew G. Barto, 2018)

Při posíleném učení se často používají algoritmy jako je Q-learning nebo Sarsa. Tyto algoritmy umožňují agentovi učit se bez učitele a maximalizovat svoji odměnu pomocí stochastického gradientového algoritmu.

3.2.3 Algoritmy strojového učení

Strojové učení je proces, při kterém počítačové systémy využívají algoritmy a statistické modely k analýze a výpočtu z velkého množství dat, aby mohly předpovídat a rozhodovat na základě nových dat. V této kapitole se zaměříme na pět základních algoritmů strojového učení: regrese, klasifikace, skládání modelů, clusterování a asociativní pravidla.

3.2.3.1 Regrese

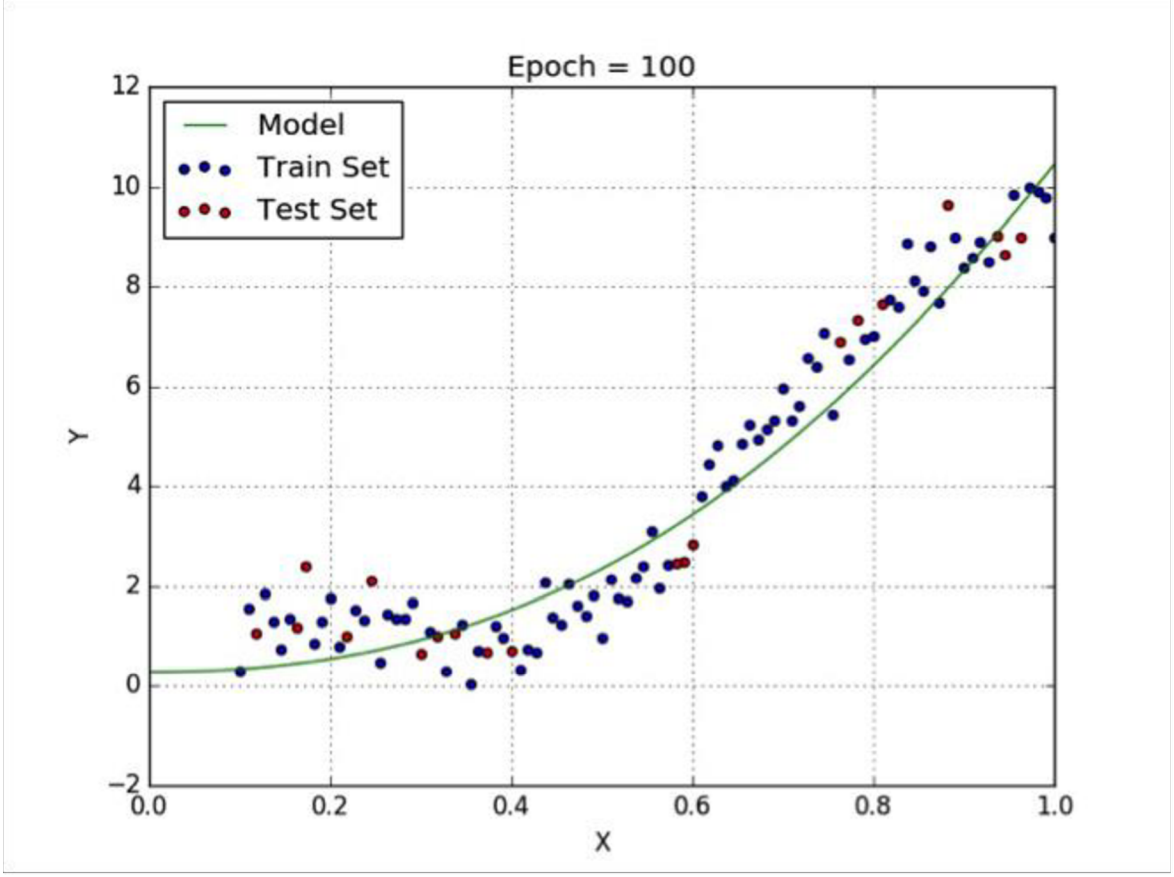
je typ algoritmu strojového učení, který se používá k predikci číselných hodnot (tzv. cílových proměnných) na základě vstupních proměnných. Regresní modely lze použít pro různé úlohy, například pro predikci ceny nemovitosti na základě počtu pokojů, lokalizace a

dalších faktorů, nebo pro predikci poptávky po produktu v závislosti na ceně, reklamních kampaních a dalších faktorech.

Nejoblíbenějším a nejpoužívanějším typem regrese je lineární regrese, která se snaží nalézt lineární závislost mezi vstupními a cílovými proměnnými. Lineární regrese se často používá jako baseline model pro porovnání s jinými typy regrese.

Příkladem může být predikce výše měsíčního příjmu na základě věku, vzdělání a zaměstnání respondentů. Vstupní proměnné jsou tedy věk, vzdělání a zaměstnání, zatímco cílovou proměnnou je výše měsíčního příjmu. Regresní model se snaží najít nejlepší lineární kombinaci vstupních proměnných, která bude co nejvíce vysvětlovat výši měsíčního příjmu.

Jako ilustrace můžeme použít následující graf. (Goodfellow, 2016)



Obrázek 6 - graf modelu regrese – Zdroj: (Vihar Kurama 2013)

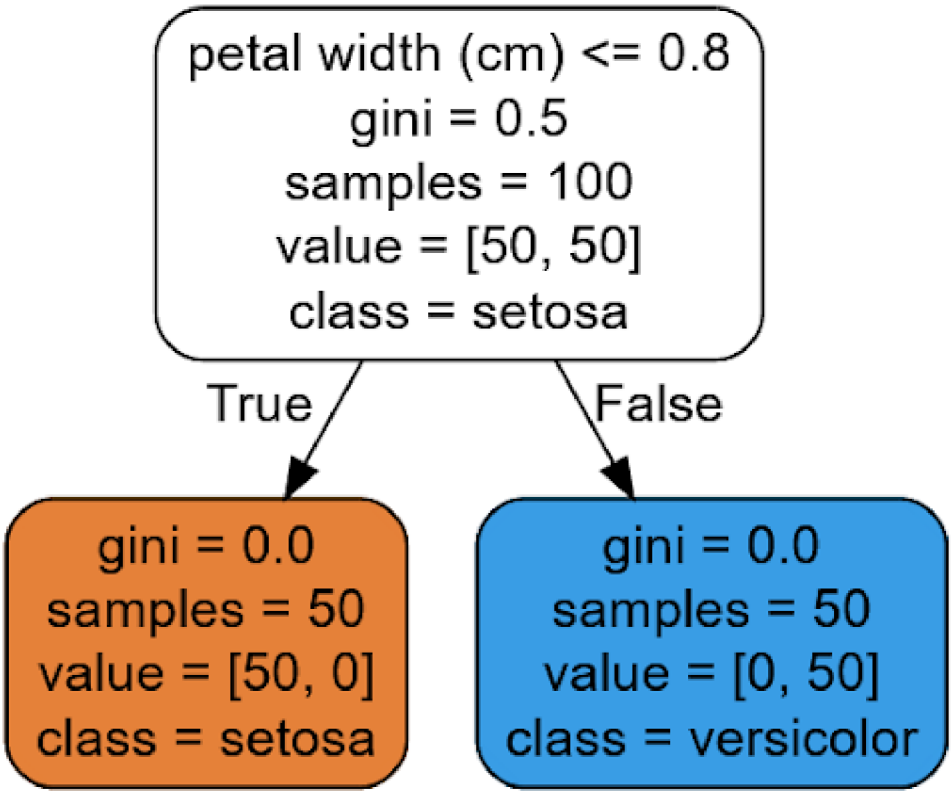
3.2.3.2 Klasifikace

patří mezi základní úlohy v oblasti strojového učení a jejím cílem je přiřadit vstupnímu datovému bodu příslušnou třídu nebo kategorii na základě naučeného modelu. Klasifikační

algoritmy se liší v závislosti na použitém modelu, který může být například rozhodovací strom, neuronová síť, k-nn algoritmus nebo SVM (Support Vector Machine).

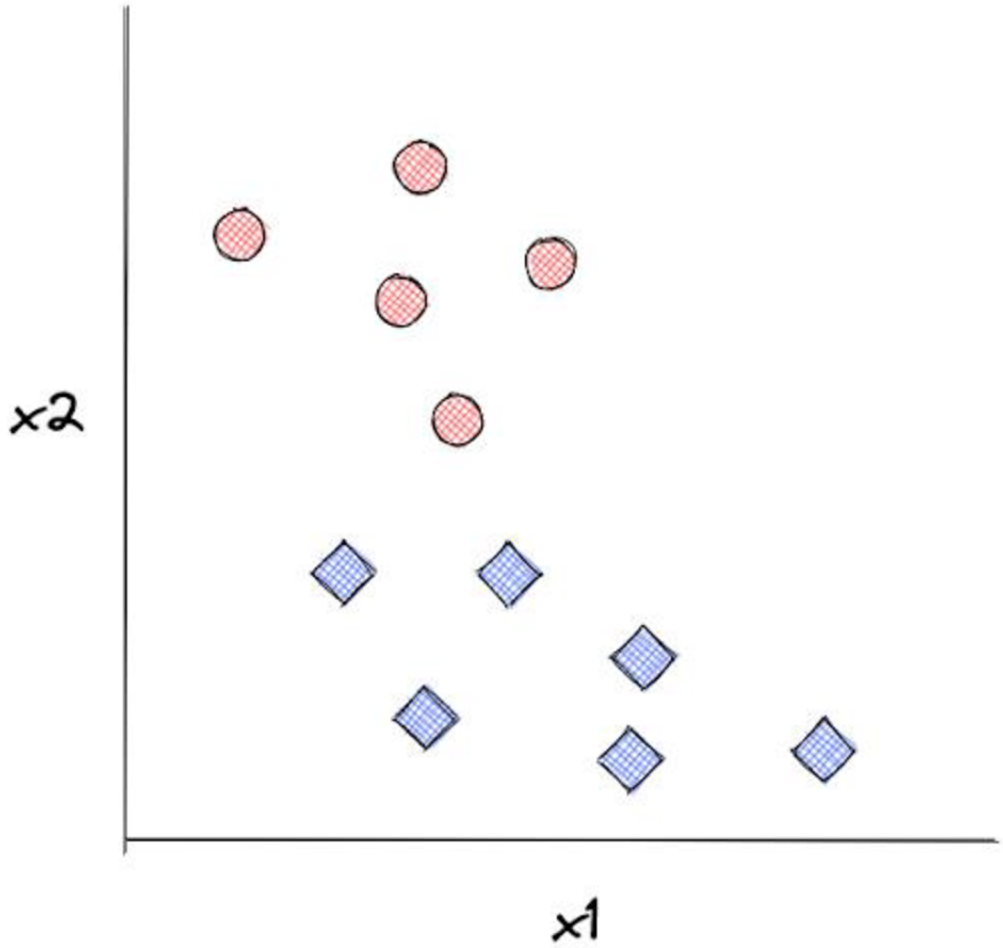
Příkladem klasifikace může být rozpoznávání psaných číslic, kdy je úkolem klasifikovat ručně psané číslice do jedné ze 10 kategorií (0-9). (T. Hastie, 2009)

Jedním z nejznámějších klasifikačních algoritmů je rozhodovací strom. Tento algoritmus vytváří stromovou strukturu rozhodování, kdy vnitřní vrcholy představují testování příznaků a listy jsou klasifikačními výstupy. Pro každý vstupní datový bod se postupně projde strom od kořene až k některému z listů, který určí výstupní třídu.



Obrázek 7 - příklad rozhodovacího stromu pro klasifikaci rostlin

Dalším klasifikačním algoritmem je k-nn (k-nearest neighbors). Tento algoritmus funguje na principu, že každý vstupní datový bod je klasifikován na základě nejbližších k sousedům v prostoru příznaků. Počet sousedů je určen parametrem k, který je předem stanoven.



Obrázek 8 - K-nn pro klasifikaci dvou tříd, Zdroj: H. Liu, J. Han, M. Kabmer,2011

Mezi další klasifikační algoritmy patří například neuronové sítě, SVM nebo Naive Bayes. Výběr konkrétního algoritmu závisí na konkrétní úloze a charakteru dat.

3.2.3.3 Skládání modelů

je technika strojového učení, která spojuje několik modelů do jednoho celkového modelu, který je schopen dosáhnout lepších výsledků než jednotlivé modely samostatně. Tato technika se používá především v oblasti hlubokého učení.
Existuje několik způsobů skládání modelů, například:
Ensembling Ensembling je technika, která spojuje výstupy několika modelů dohromady. Tento postup se používá k dosažení lepších výsledků při klasifikaci a regresi. Jedním z příkladů ensemblingu je metoda bagging, která spojuje výsledky několika rozhodovacích stromů.

Transfer learning Transfer learning je technika, která spojuje výsledky modelů využitých při trénování jiných úloh a používá je pro novou úlohu. Tato technika se používá především v oblasti hlubokého učení.

Stacking spojuje výstupy více modelů, přičemž výstupy jsou použity jako vstup pro nový model. Tento nový model pak produkuje konečný výstup. Používá se především při klasifikaci a regresi.

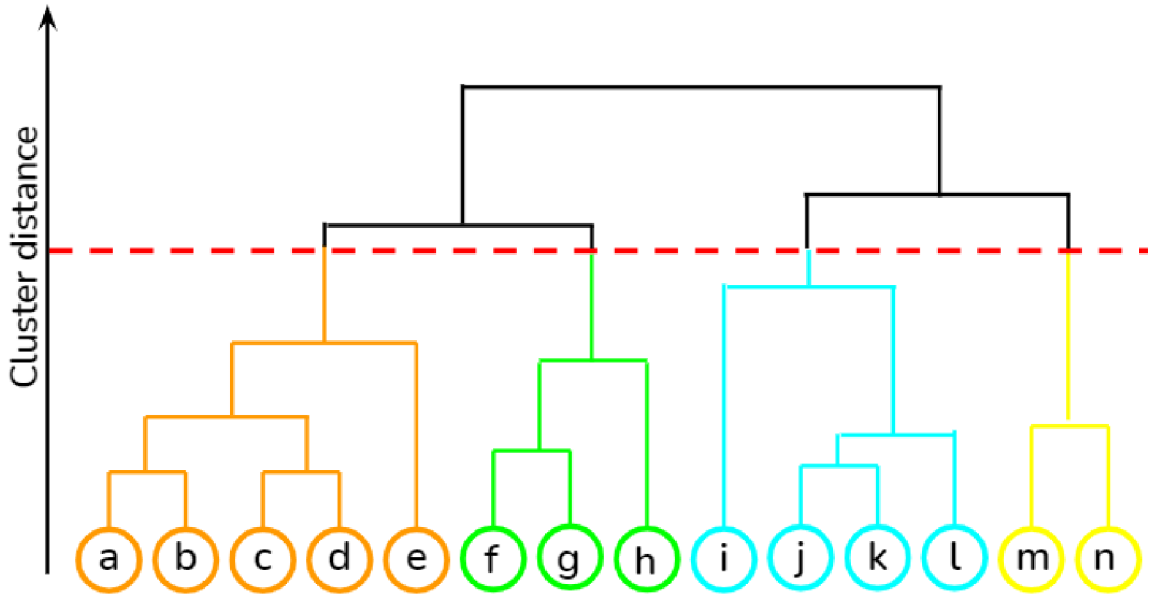
Příkladem skládání modelů může být kombinace klasifikátorů v ensemblingu. Například můžeme použít několik klasifikátorů, jako jsou rozhodovací stromy, náhodné lesy a neuronové sítě, a následně kombinovat jejich výstupy pro dosažení lepší přesnosti klasifikace. (Bishop, 2006)

3.2.3.4 Clustering

je metoda strojového učení, která se používá k rozdělení dat do skupin (clusters) na základě jejich podobnosti. Skupiny jsou vytvářeny tak, aby objekty v jedné skupině byly si co nejvíce podobné, zatímco objekty v jiných skupinách byly si co nejvíce odlišné. Tento proces umožňuje nalézt vzorce a struktury v datech, které by jinak mohly být skryty.

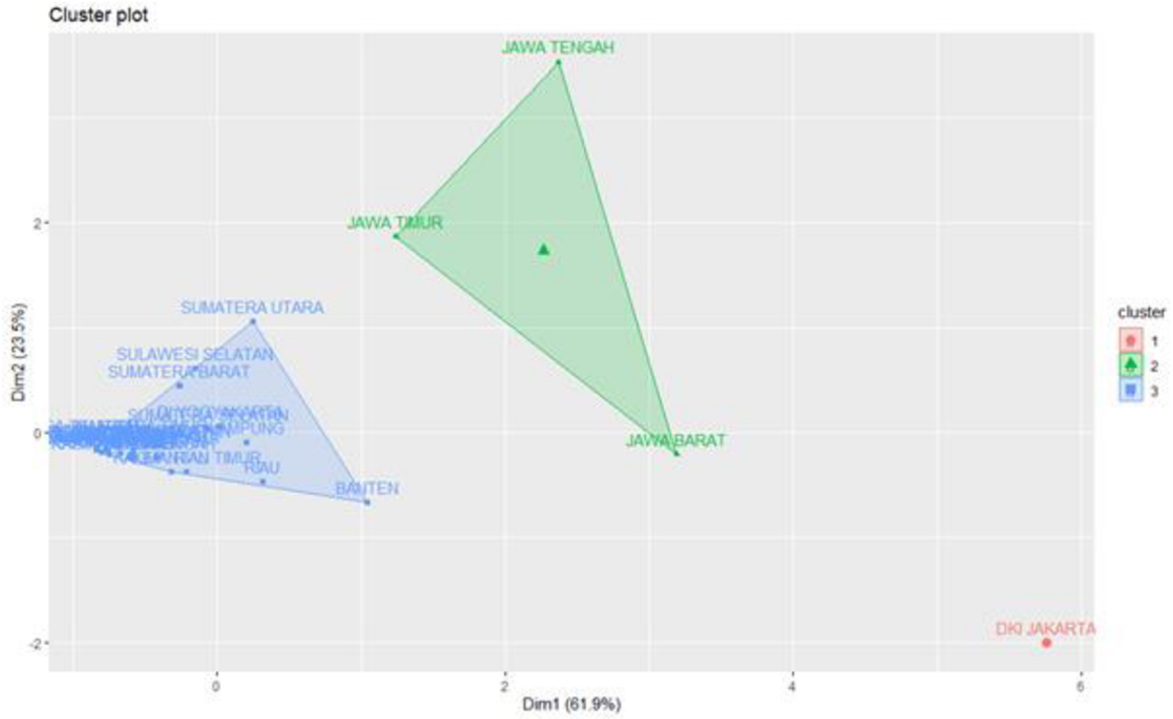
Existují různé algoritmy clusteringu, které se liší výpočetní složitostí a přesností. Zde uvedeme dva základní přístupy:

Hierarchický clustering - Tento přístup se řadí ke stromovým metodám, kde se postupně skládají menší skupiny do větších a větších skupin, dokud není dosaženo jedné velké skupiny obsahující všechny objekty. Existují dva základní typy hierarchického clusteringu: aglomerativní a divisive. Aglomerativní přístup začíná s jednotlivými objekty a postupně je slévá do větších skupin. Divisive přístup začíná s celkovou skupinou a postupně ji rozděluje na menší skupiny.



Obrázek 9 - hierarchický clustering

Nehierarchický clustering - Tento přístup se řadí ke k-means metodám, kde se určuje pevný počet skupin a následně se objekty přiřazují k nejbližší skupině na základě vzdálenosti mezi nimi.



Obrázek 10 - nehierarchický clustering

Příkladem použití clusteringu může být klasifikace zákazníků do skupin na základě podobnosti jejich chování nebo preference produktů. Dalším příkladem může být

identifikace skupin pacientů s podobnými charakteristikami pro účely výzkumu nebo vytváření personalizované medicíny. (Jain, 1999 str. 264)

3.2.3.5 Asociativní pravidla

Asociativní pravidla jsou dalším typem algoritmů strojového učení, které se využívají pro hledání vztahů mezi daty a identifikaci závislostí mezi různými proměnnými. Tyto algoritmy jsou využívány především v oblasti analýzy velkých datových sad.

Asociativní pravidla jsou založena na vztahu mezi podmínkou (antecedentem) a důsledkem (konsekventem). Podmínka a důsledek jsou spojeny pomocí pravidla, které může být definováno jako "pokud A, pak B". Cílem algoritmu je najít taková pravidla, která mají vysokou pravděpodobnost, že jsou splněna.

Jedním z nejznámějších příkladů asociativních pravidel jsou Apriori pravidla. Tyto pravidla jsou používány pro analýzu košů s nákupy a zjištění, které produkty jsou často kupovány spolu. Například, pokud si zákazník kupuje chléb, je pravděpodobné, že si koupí také máslo. (Han, 2011)

3.2.4 Deep Learning (Neuronové sítě)

je forma strojového učení založená na neuronových sítích, která se snaží napodobit fungování lidského mozku. Neuronové sítě jsou sestaveny z mnoha vzájemně propojených neuronů, které přijímají vstupní data, zpracovávají je a generují výstupy.

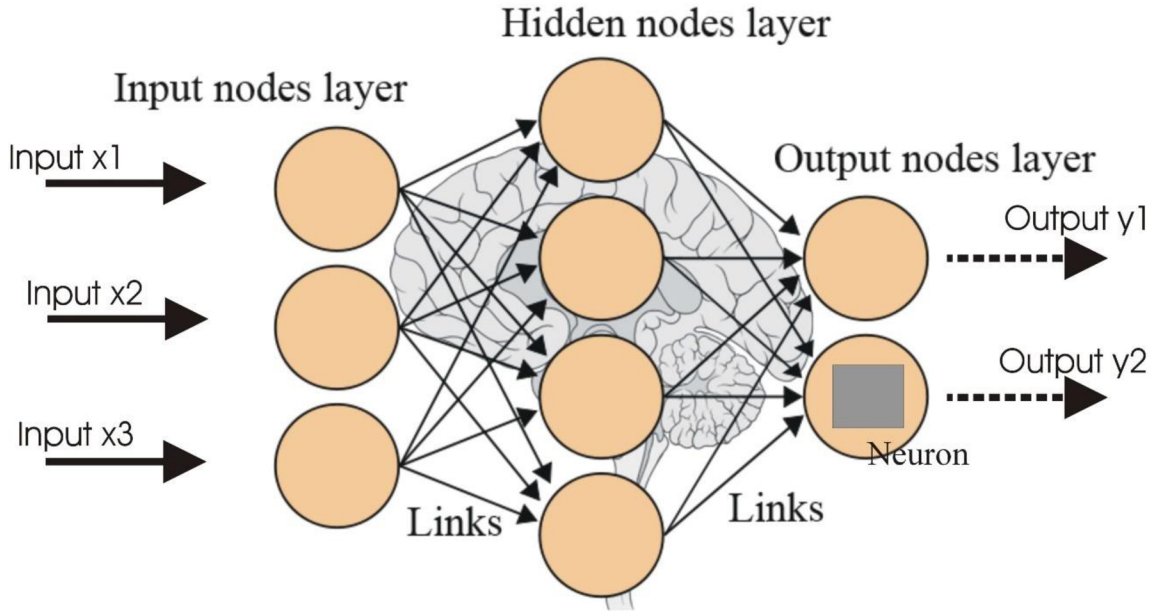
Deep learning se vyznačuje tím, že neuronové sítě jsou velmi hluboké, tzn. obsahují mnoho vrstev. Tyto vrstvy jsou schopny automaticky extrahovat složité a abstraktní vlastnosti z dat, což umožňuje modelům deep learningu dosáhnout vynikajících výsledků při řešení komplexních úloh.

Neuronové sítě se skládají z mnoha vrstev neuronů, které jsou propojeny váhami. Neuron v každé vrstvě vypočítává váženou sumu svých vstupů a aplikuje na ni aktivační funkci, aby generoval výstup.

Deep learning využívá několik typů neuronových sítí, jako jsou konvoluční neuronové sítě (CNN), rekurentní neuronové sítě (RNN) a generativní modely.

Architektura neuronových sítí se skládá z několika vrstev neuronů, které jsou navzájem propojeny. Základními vrstvami jsou vstupní, skryté a výstupní vrstvy.

- Vstupní vrstva je první vrstvou neuronové sítě a přijímá vstupní data, jako jsou obrázky, zvuky nebo text. Data jsou převedena na číselný formát, aby je bylo možné zpracovat.
- Skryté vrstvy jsou vrstvy neuronů, které přijímají vstupní data z předchozí vrstvy a generují výstupy, které jsou předány do další vrstvy. Tyto vrstvy jsou nazývány skryté, protože nejsou viditelné při předávání vstupu nebo výstupu.
- Výstupní vrstva je poslední vrstvou neuronové sítě a generuje výstup na základě výstupů z předchozí vrstvy. Tento výstup může být různých typů, například klasifikace nebo predikce. V případě klasifikace se výstupní vrstva skládá z několika neuronů, kde každý neuron reprezentuje jednu třídu. Každý neuron v této vrstvě generuje hodnotu, která určuje pravděpodobnost, že vstupní data patří do příslušné třídy. V případě predikce může být výstupní vrstva tvořena jedním nebo více neurony, kde každý neuron reprezentuje jednu výstupní hodnotu.



Obrázek 11 - jednoduchá neuronová síť se třemi vrstvami

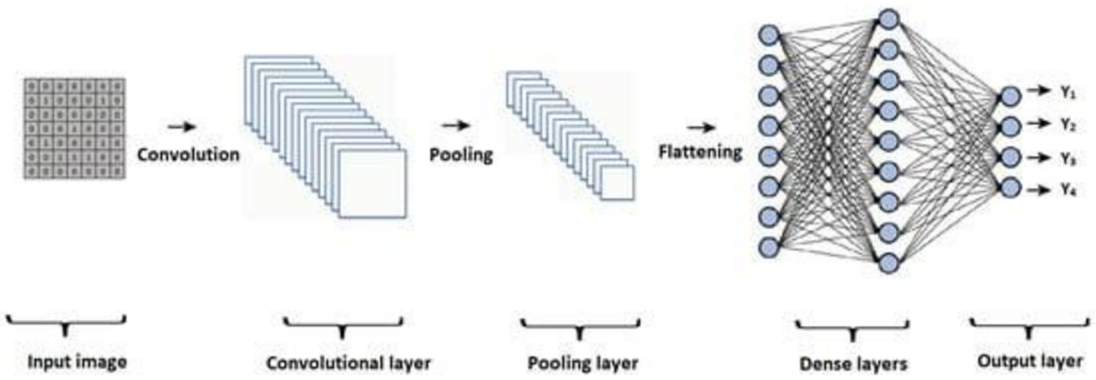
Ilustrace níže ukazuje příklad architektury neuronové sítě se třemi vrstvami: vstupní vrstvou, skrytou vrstvou a výstupní vrstvou. (Goodfellow, 2016)

3.2.4.1 Konvolunční neuronové sítě

Konvoluční neuronové sítě (CNN) jsou druhem neuronových sítí, které se často používají pro zpracování obrazových a jiných druhů vícedimenzionálních dat. Konvoluční vrstvy jsou hlavním prvkem CNN a využívají konvoluce pro extrakci rysů z vstupních dat.

Konvoluce je matematická operace, která se používá k transformaci signálu na jiný signál. V kontextu konvolučních neuronových sítí se konvoluce používá k aplikaci filtrů na vstupní data s cílem extrahovat rysy. Filtry jsou matice o určité velikosti, které se aplikují na vstupní data postupně. Výsledkem aplikace filtru na danou oblast vstupních dat je jedna hodnota, která tvoří jeden prvek výstupní matice. Opakováním této operace na celém vstupním obraze se vytváří matice rysů, které jsou následně zpracovávány dalšími vrstvami v konvoluční neuronové síti.

Konvoluční neuronové sítě se skládají z řady vrstev. Obvyklá architektura konvoluční neuronové sítě se skládá z několika konvolučních vrstev, následovaných max pooling vrstvou a lineární vrstvou na konci sítě. Konvoluční vrstvy slouží k extrakci rysů, zatímco max pooling vrstva snižuje rozměr výstupu z konvoluční vrstvy a tím zlepšuje výkon neuronové sítě. Lineární vrstva slouží k třídění nebo predikci dat. (LeCun, 2015 stránky 436-444)



Obrázek 12 - konvoluční neuronová síť

Ilustrace níže ukazuje příklad architektury konvoluční neuronové sítě. Na konci sítě se nachází několik plně propojených vrstev, které slouží ke klasifikaci nebo regresi výsledků. Tyto vrstvy mohou být kombinovány s různými aktivačními funkcemi, jako je například ReLU (Rectified Linear Unit), která se často používá v konvolučních sítích.

Konvoluční neuronové sítě se staly velmi populární pro úlohy zpracování obrazu a dosahují v mnoha případech state-of-the-art výsledků. Mezi nejznámější architektury patří například VGG, ResNet, Inception a MobileNet.

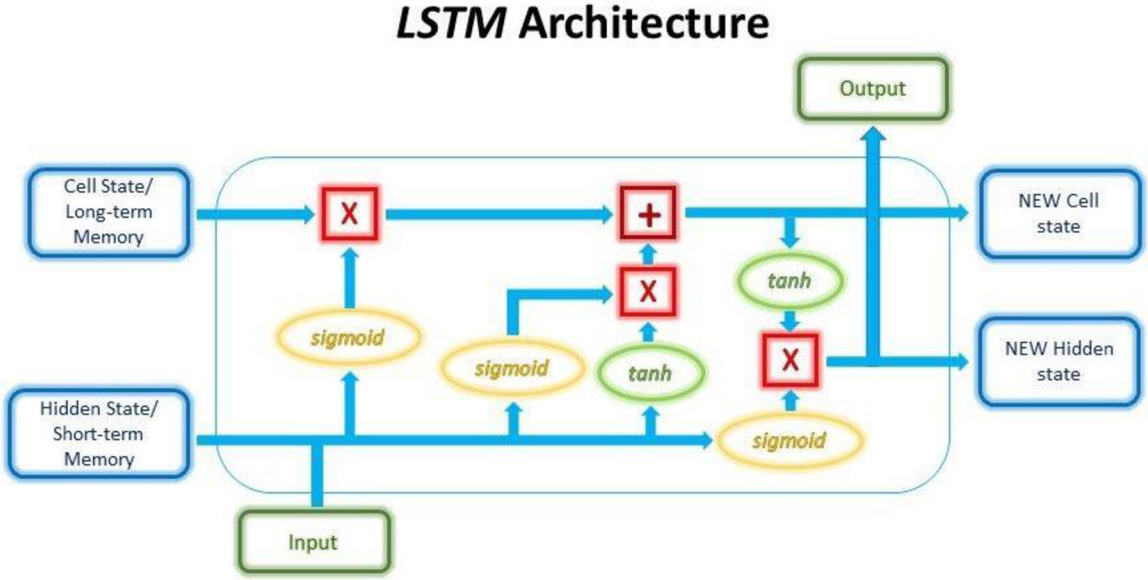
3.2.4.2 Rekurentní neuronové sítě

jsou typem neuronových sítí, které jsou vhodné pro zpracování dat s časovou závislostí, například pro analýzu časových řad, jazykové modelování nebo překlad textu. Na rozdíl od konvolučních sítí mají RNN paměťovou funkci, která jim umožňuje uchovat informace o dřívějších vstupech a použít je při zpracování aktuálního vstupu.

Základním prvkem RNN jsou rekurentní neurony, které jsou propojeny tak, aby výstup jednoho neuronu byl součástí vstupu dalšího neuronu v následujícím kroku. Tento propojený řetězec neuronů umožňuje RNN pracovat s proměnnou délkou vstupních sekvencí.

RNN jsou obecně tvořeny jedním nebo více rekurentními vrstvami, které obsahují rekurentní jednotky, a výstupní vrstvou, která může být tvořena například lineární vrstvou pro regresi nebo vrstvou softmax pro klasifikaci. Mezi typické rekurentní jednotky patří LSTM (Long-Short Term Memory) a GRU (Gated Recurrent Unit).

Jedním z nejznámějších typů RNN jsou Long Short-Term Memory Networks (LSTM). Tyto sítě jsou navrženy tak, aby lépe zachytily dlouhodobé závislosti v datech, které jsou pro běžné RNN obtížné. Každý neuron v LSTM má tři vstupy: vstupní signál, informaci o minulém stavu a informaci o minulé paměti. Výstupem neuronu je nový stav a nová paměť, které jsou předány dalším neuronům v řetězci.



Obrázek 13 - architektura LSTM

LSTM a GRU jsou navrženy pro řešení problému s tzv. "zmizením a explozí gradientů", což je častý problém při trénování RNN. Tyto jednotky v sobě mají speciální mechanismy,

které jim umožňují se rozhodovat, které informace si mají pamatovat a které zapomenout, a tím řeší zmíněný problém.

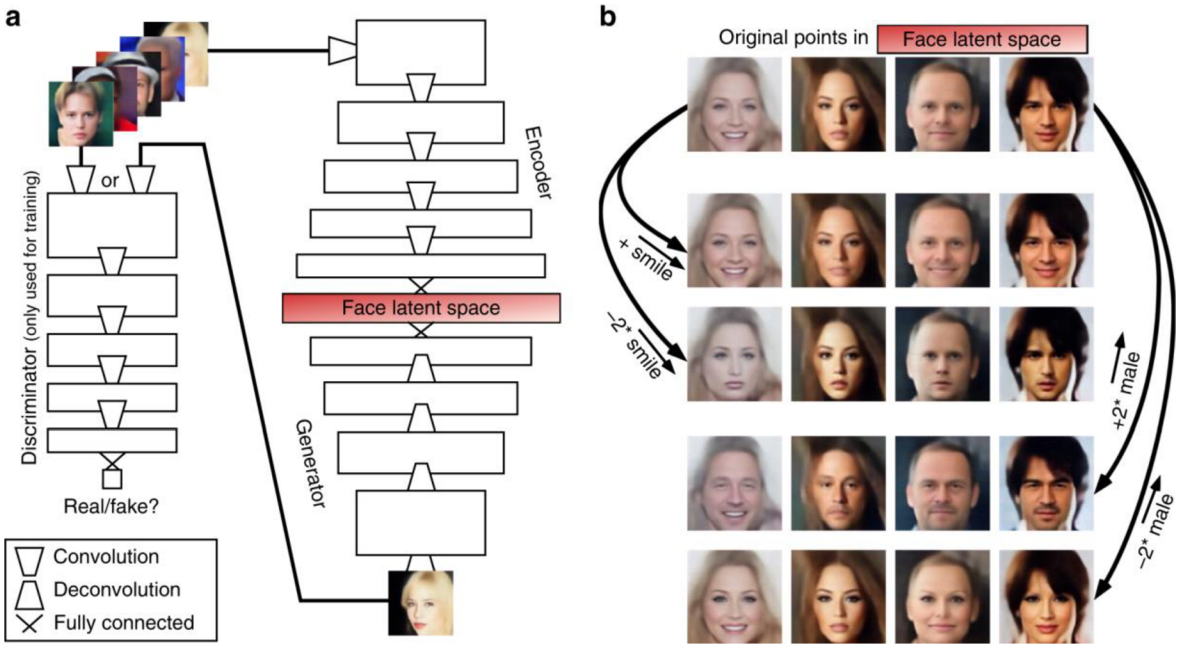
Příkladem použití RNN může být například předpověď slova v textu na základě předchozích slov, automatický překlad, rozpoznávání řeči, generování hudby a mnoho dalších aplikací. (Goodfellow, 2014)

3.2.4.3 Generativní modelování

je obecná metoda strojového učení, která se snaží naučit se pravděpodobnostní rozdělení datového prostoru a na jeho základě generovat nová data. Tato metoda se používá pro generování nových obrázků, zvuků, textů a dalších druhů dat.

Generativní modely mohou být založeny na různých algoritmech, například na neuronových sítích nebo na Bayesovských sítích. Tyto modely se učí pravděpodobnostní rozdělení datového prostoru na základě trénovacích dat a poté jsou schopny generovat nová data, která jsou podobná těm trénovacím.

Příkladem generativního modelování je například generování obrázků. Neuronové sítě mohou být naučeny rozpoznávat vzory v obrazech a na základě těchto vzorů pak generovat nové obrázky, které jsou podobné těm trénovacím. K tomu využívají takzvané generativní modely, které jsou schopné generovat náhodné vektory, které jsou poté transformovány na výstupní obrázky. (Bishop, 2006)



Obrázek 14 - příklad generování tváří pomoci Generativního modelu

Na obrázku níže je příklad generování různých tváří pomocí generativního modelu.

3.2.5 Prognostické metódy a predikce

Predikce nebo prognóza je proces odhadování budoucího stavu nebo vývoje na základě dostupných informací a dat. Prognostické metody jsou nástroje, které nám umožňují vypočítat a odhadnout budoucí vývoj nebo chování daného jevu nebo systému. V dnešní době je predikce a prognóza velmi důležitá v mnoha odvětvích, jako je například ekonomie, finančnictví, marketing, doprava, energetika a další.

Existuje mnoho různých přístupů a metod, které lze použít k predikci a prognóze. Mezi nejznámější patří statistické metody, strojové učení a umělá inteligence. Tyto metody mohou být použity samostatně nebo v kombinaci a záleží na konkrétní situaci a úkolu, který chceme řešit.

Prognostické modelování může být velmi složité a zahrnuje mnoho faktorů, které mohou ovlivnit výsledky predikce. Je důležité mít k dispozici dostatečné množství relevantních dat a správně je analyzovat a zpracovat. Je také důležité vybrat správnou metodu pro konkrétní úkol a být obezřetní při interpretaci výsledků.

Mezi nejčastěji používané predikční metody patří ARIMA a VAR modely, které jsme si již v předchozích kapitolách rozebrali. Tyto modely dokážou pracovat s časovými řadami a predikovat budoucí hodnoty na základě historických dat.

Dalšími prognostickými metodami jsou například neuronové sítě, rozhodovací stromy, regresní analýza a další. Tyto metody umožňují pracovat s komplexními datovými strukturami a vztahy mezi proměnnými.

Při predikci je důležité dbát na správnou volbu proměnných, které budou mít vliv na predikovanou hodnotu, a na správnou volbu vstupních dat, které budou použity k trénování modelu.

3.2.5.1 Lineární regrese

regrese je metoda pro určení vztahu mezi dvěma proměnnými pomocí lineárního modelu. Jedná se o základní metodu pro predikci a využívá se například v ekonomii k predikci vývoje trhu nebo ceny produktů.

Příklad:

Mějme data z průměrného denního počtu návštěv na webových stránkách za poslední měsíc. Snažíme se predikovat počet návštěv na zítřejší den.

Tabulka 2 - příklad lineární regrese

DEN	POČET NÁVŠTĚV
1	200
2	220
3	250
4	230
5	240
6	270
7	280
8	300
9	320
10	350
11	370
12	390
13	400
14	420
15	450
16	480
17	500
18	530
19	550
20	580
21	600
22	630
23	650
24	670
25	700
26	720
27	750
28	780
29	800
30	830

Použijeme lineární regresi k vytvoření modelu predikce počtu návštěv na zítřejší den.
Výsledkem je model predikce počtu návštěv na zítřejší den. Pokud například předpokládáme, že bude zítřejší den stejně populární jako den 22., pak můžeme předpovědět, že bude mít webová stránka přibližně 632 návštěv. (T. Hastie, 2009 stránky 485-493)

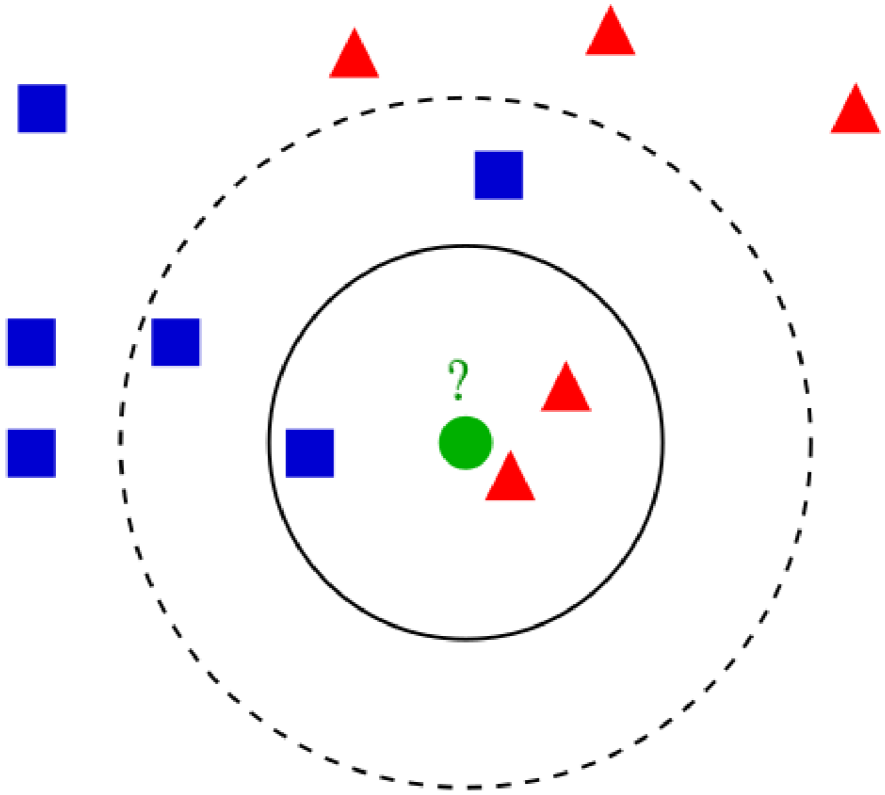
3.2.5.2 K-Means clustering

je metoda unsupervised učení, která se používá pro shlukování (clustering) dat. Jeho hlavním cílem je rozdělit n datových bodů do k shluků tak, aby každý bod byl přiřazen ke shluku s nejbližším středem. Tento algoritmus pracuje v několika krocích:
Náhodně se vyberou počáteční ke středů shluků.
Každý datový bod je přiřazen ke shluku, jehož střed je nejblíže (podle eukleidovské vzdálenosti).
Střed každého shluku se přepočítá jako průměr všech bodů v tomto shluku.
Proces přiřazování bodů ke shlukům a přepočítávání středů se opakuje, dokud se středy neustálí a nedojde ke konvergenci algoritmu.
Konečný výsledek k-means clustering je tedy přiřazení každého datového bodu ke shluku s nejbližším středem. Tato metoda může být velmi užitečná pro vytváření segmentů pro marketingové účely, kategorizaci zákazníků, identifikaci anomálií nebo hledání podobností mezi různými datovými body. (Bishop, 2006 stránky 422-429)

3.2.5.3 K-Nearest neighbors

je jednoduchá a intuitivní metoda strojového učení pro klasifikaci a regresi. Tato metoda se řadí mezi tzv. "Lazy learning" metody, což znamená, že model je trénován až v době predikce, tedy využívá pouze data, která má k dispozici v daném okamžiku. K-NN je založena na předpokladu, že podobné příklady jsou blízko sebe v prostoru příznaků. V K-NN algoritmu se klasifikuje nebo regresuje na základě hodnoty K, což určuje, kolik nejbližších sousedů bude bráno v úvahu při predikci.
Při klasifikaci se používá tzv. "většinové hlasování", tedy třída, ke které náleží většina nejbližších sousedů, se stane predikovanou třídou. Při regresi se používá průměrná hodnota nejbližších sousedů jako predikovaná hodnota.

Příklad: Předpokládejme, že máme data set se dvěma příznaky, např. výška a váha, a chceme klasifikovat osoby jako "štíhlé" nebo "obézní". Pokud máme novou osobu s výškou 180 cm a váhou 80 kg, použijeme K-NN s K=5 a najdeme 5 nejbližších sousedů v data setů. Pokud z těchto 5 sousedů 3 jsou klasifikováni jako "štíhlí" a 2 jako "obézní", nová osoba bude klasifikována jako "štíhlá". (Scikit-learn)



Obrázek 15 - K-NN algoritmus

3.2.5.4 Klasifikace pomocí neuronové sítě (Regrese)

Klasifikace pomocí neuronových sítí spočívá v tom, že se trénování neuronové sítě na data setů, kde jsou vstupní data klasifikována do několika kategorií. Síť se učí, jakým způsobem propojení neuronů a jejich vah vede k co nejpřesnější klasifikaci dat.

Příkladem může být klasifikace různých druhů květin podle jejich parametrů. Neuronová síť se učí, jakým způsobem propojit jednotlivé vstupy, aby byla schopna rozeznat jednotlivé druhy květin.

Regrese pomocí neuronových sítí spočívá v tom, že se neuronová síť učí predikovat numerickou hodnotu na základě vstupních dat. Síť se učí, jakým způsobem propojení neuronů a jejich vah vede k co nejpřesnější predikci dat.

Příkladem může být predikce ceny nemovitosti na základě různých faktorů, jako jsou velikost domu, počet pokojů, poloha atd. Neuronová síť se učí, jakým způsobem propojit

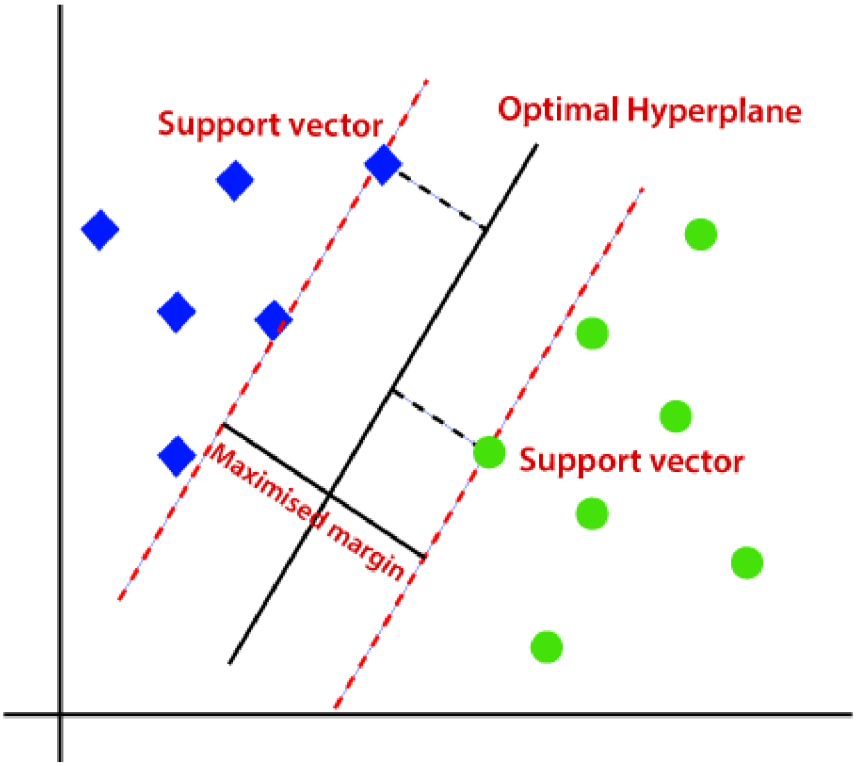
jednotlivé vstupy, aby byla schopna predikovat cenu nemovitosti co nejpřesněji. (Goodfellow, 2016 stránky 145-176)

3.2.5.5 Super Vector Machines (SVM)

se často používá jako predikční algoritmus, zejména pro binární klasifikaci, tedy rozdělení datových bodů do dvou tříd. SVM se snaží najít hyper-rovinu v prostoru, která co nejlepším způsobem odděluje tyto třídy od sebe. Při trénování SVM se vybírají takové vzorky (tzv. support vectors), které leží nejblíže této hyper-rovině.

SVM může být použita i pro multiklasovou klasifikaci pomocí metody one-vs-all, kde se pro každou třídu vytvoří SVM a výsledek klasifikace se určí podle toho, která třída má nejvyšší hodnotu výstupu SVM.

Příkladem použití SVM může být klasifikace spamu a ne-spamu v e-mailové schránce. Pro trénování SVM se použijí například e-maily, u kterých je předem známo, zda jsou spam nebo ne. Tyto e-maily jsou reprezentovány jako vektory příznaků, kde každý prvek představuje určitý atribut (např. počet slov, výskyt určitých slov atd.). SVM se poté natrénuje na těchto vektorech a při klasifikaci nových e-mailů se použije natrénovaná hyper-rovina. (Bishop, 2006 stránky 129-154)



Obrázek 16 - SVM algoritmus

Ilustrační obrázek SVM může ukázat, jak SVM hledá hyper-rovinu, která nejlépe odděluje třídy.

3.2.5.6 Autoregressive Integrated Moving Average (ARIMA)

Existují různé typy časových řad, základní rozdělení zahrnuje stacionární a nestacionární časové řady. Stacionární časová řada je taková, která má konstantní průměr a rozptyl v čase a nezávisí na čase. Na druhé straně nestacionární časové řady mají průměr a/nebo rozptyl, které se mění v průběhu času. Dále se mohou časové řady dělit například na periodické a aperiodické, lineární a nelineární, deterministické a stochastické apod.
Při modelování časových řad se používají různé metody, včetně statistické analýzy, neuronových sítí a dalších strojových učení. Při analýze časových řad je obvykle nutné určit typ řady a přizpůsobit model tak, aby co nejlépe odpovídal datům.

jsou jednou z nejčastěji používaných metod pro modelování časových řad. Tyto modely jsou založeny na kombinaci autoregresního (AR) a krokového průměru (MA) modelu, a jsou vhodné pro modelování časových řad s nekonstantním průměrem, sezónností nebo trendem.
ARIMA modely jsou definovány třemi parametry: p, d a q, kde:

- p (pořadí autoregresního členu) určuje počet předchozích hodnot, které jsou zahrnuty v modelu.
- d (stupeň integrace) určuje počet diferencování potřebných k dosažení stacionarity v časové řadě.
- q (pořadí krokového průměru) určuje počet předchozích chyb, které jsou zahrnuty v modelu.

ARIMA modely jsou často doplňovány o sezónní komponentu, což se označuje jako SARIMA (Seasonal ARIMA) modely. Tyto modely zahrnují sezónní diferencování a sezónní autoregresní a krokové průměrné členy. (Brockwell, 2016 stránky 201-235)

Pro implementaci ARIMA modelů lze použít různé statistické nástroje, jako je například Python knihovna Statsmodels nebo software R.

3.2.5.7 Vector Autoregression modely (VAR)

jsou statistické modely, které umožňují analýzu závislosti mezi více proměnnými v čase. Tyto modely se často používají pro predikci časových řad. V modelu se vyskytuje více proměnných, které jsou vzájemně závislé, což umožňuje zohlednění vlivu jedné proměnné na jinou.

Jedním z příkladů použití VAR modelů může být analýza vztahu mezi objemem prodaného zboží a výdaji na reklamu. Na základě této analýzy může být vytvořen model pro predikci budoucího prodeje na základě výdajů na reklamu.

Jeden z nejčastěji používaných přístupů pro odhadování parametrů VAR modelů je metoda maximální věrohodnosti (maximum likelihood method). Pro odhadování parametrů VAR modelů je třeba mít k dispozici dostatečně velký datový soubor.

Jeden z nevýhod VAR modelů je, že pokud jsou proměnné navzájem silně závislé, může být model nestabilní a může docházet k velkým chybám v predikci.

Na rozdíl od VAR modelů, které umožňují analýzu závislosti mezi více proměnnými, ARIMA modely se zaměřují na analýzu jedné časové řady. ARIMA modely umožňují odhadování trendu, sezónnosti a náhodné složky v časové řadě.

Pro odhadování parametrů ARIMA modelů se také často používá metoda maximální věrohodnosti (maximum likelihood method). Jeden z nevýhod ARIMA modelů je, že se často setkávají s problémy, pokud je v časové řadě příliš mnoho náhodných vlivů.

4 Analytická část

4.1 Analýza a komparace možností vývoje

Po shrnutí teoretických poznatků a úvodu do tématu následuje analytická část práce. V této části budou představeny metody učení a predikce, které budou následně použity pro návrh aplikace pro automatizované obchodování na burze s napojením na Binance API a automatické obchodování trade botem pro prodej na určité úrovni. Dále bude v analytické části prezentována analýza a predikce časových řad jednotlivých kryptoměn, která bude založena na metodách popsaných v předchozí kapitole.

Další kapitolou bude "Porovnání prognostických metod pro nejlepší využití k navrhovanému řešení", která se zaměří na porovnání různých metod predikce časových řad a určení nejvhodnější metody pro použití v navrhované aplikaci.

Poslední kapitolou analytické části bude analýza vhodného jazyka a knihoven s využitím strojového učení/deep learning pro použití na vývoj navrhovaného řešení. V této kapitole budou představeny různé jazyky a knihovny a bude provedeno porovnání jejich výhod a nevýhod pro využití v navrhované aplikaci.

V rámci této práce bude také vytvořen stručný popis navrhované aplikace a vysvětlení jejího fungování.

4.1.1 Analýza a predikce časový řad

Pro porovnání metod budou použity metriky jako MAE (mean absolute error), RMSE (root mean square error) a MAPE (mean absolute percentage error). Tyto metriky nám poskytnou představu o tom, jak dobře se modely přizpůsobují datům a jak přesné jsou jejich predikce.

Výsledky analýzy a porovnání metod budou prezentovány v tabulce, která bude zahrnovat metriky pro každou metodu a pro každou kryptoměnu. Bude také prezentována vizualizace časových řad a predikcí pro každou kryptoměnu a pro každou metodu.

Pro tuto analýzu budou použity historické ceny a objemy kryptoměn, které budou získány z datových zdrojů jako je například Binance API a další.

V následující tabulce porovnávám různé metody používané pro predikci časových řad v kontextu kryptoměn:

Tabulka 3 - provnání metod z hlediska vlastností

Metoda	Přesnost predikce	Flexibilita	Složitost implementace	Požadavky na data
ARIMA	Vysoká	Nízká	Střední	Lineární
LSTM	Vysoká	Vysoká	Vysoká	Sekvenční
FB Prophet	Vysoká	Střední	Nízká	Časová řada

Z výše uvedené tabulky vyplývá, že metody LSTM a FB Prophet jsou obecně lepší pro predikci časových řad v kontextu kryptoměn než ARIMA model. Metoda LSTM má vysokou přesnost predikce a vysokou flexibilitu, ale její implementace je složitá a vyžaduje sekvenční data. Na druhé straně, metoda FB Prophet má střední flexibilitu, nízkou složitost implementace a vyžaduje pouze časovou řadu jako vstup. Tato metoda by mohla být vhodná pro jednodušší aplikace, kde není potřeba tak přesná predikce jako u metody LSTM.

Je důležité poznamenat, že volba nejlepší metody pro predikci časových řad závisí na specifických potřebách aplikace a charakteru dat. Proto by měla být provedena podrobná analýza a porovnání různých metod pro nalezení nejvhodnějšího řešení pro konkrétní problém.

Pro každou kryptoměnu jsme použili stejný datový soubor, který obsahoval historické ceny kryptoměn až po současnost. Data byla dále rozdělena na trénovací a testovací sady, kde trénovací sada zahrnovala 80% dat a testovací sada obsahovala zbývajících 20% dat.

Metoda ARIMA (Autoregressive Integrated Moving Average) je lineární model pro analýzu a predikci časových řad. LSTM (Long Short-Term Memory) je hluboká neuronová síť, která může zpracovávat časové řady s nepravidelnými vlnitými vzory a nepředvídatelnými fluktuacemi. Facebook Prophet je open source knihovna pro predikci časových řad vyvinutá týmem Facebook Research. (Hyndman, 2018.)

V naší analýze jsme porovnávali úspěšnost těchto tří metod v predikci cen dvou kryptoměn – Bitcoin (BTC) a Ethereum (ETH). Pro každou metodu jsme provedli predikci na 30 dnů dopředu a porovnali ji s reálnými cenami kryptoměn v testovací sadě. Poté jsme vypočítali několik statistických metrik, abychom mohli porovnat úspěšnost každé metody.

Výsledky naší analýzy ukázaly, že Facebook Prophet byla nejpřesnější metodou pro predikci cen kryptoměn, když dosáhla nejnižších hodnot pro chybu predikce (RMSE) a vysokou hodnotu koeficientu determinace (R-squared). Na druhém místě se umístil LSTM a ARIMA byla nejméně úspěšnou metodou, když dosáhla nejvyšší hodnoty RMSE a nejnižší hodnoty

Tabulka srovnání metod:

Tabulka 4- výsledky RMSE a R-squared

Metoda	*BTC RMSE*	*BTC R-squared*	*ETH RMSE*	*ETH R-squared*
ARIMA	1775.45	0.778	71.42	0.935
LSTM	1269.86	0.868	44.62	0.973
Facebook Prophet	771.68	0.947	26.75	0.989

Bitcoin: https://coinmarketcap.com/currencies/bitcoin/historical-data/
Ethereum: https://coinmarketcap.com/currencies/ethereum/historical-data/

V Pythonu, konkrétně pro ARIMA v knihovně statsmodels, pro LSTM v knihovně Keras a pro Facebook Prophet v knihovně prophet. (Tsantekidis, 2018. stránky 1-17)
V tabulce porovnání metod jsou uvedeny výsledky pro obě kryptoměny, kde můžeme vidět hodnoty RMSE a R-squared pro každou metodu. Hodnota RMSE udává střední hodnotu rozdílu mezi predikovanou cenou a reálnou cenou v testovací sadě. Vyšší hodnota RMSE znamená, že predikce byla méně přesná. Hodnota R-squared je koeficient determinace, což udává, jak dobře se predikovaná data shodují s reálnými daty. Vyšší hodnota koeficientu determinace (R-squared) znamená, že predikce se shoduje s reálnými daty lépe než v případě nižší hodnoty R-squared. R-squared udává, jak velkou část variace v datech vysvětluje model. Hodnota R-squared se pohybuje v rozmezí 0 až 1, kde hodnota 1 znamená, že model dokonale vysvětluje variabilitu v datech. Tedy ve srovnání s nižší hodnotou R-squared, vyšší hodnota R-squared znamená, že predikce byla úspěšnější a model vysvětluje větší část variability v datech. Literatura, která bude použita pro tuto kapitolu, zahrnuje publikace a články věnované predikci časových řad a kryptoměnám. (Jiang, 2021.)

4.1.2 Porovnání prognostických metod

Pro analýzu budeme používat časové řady kryptoměn Bitcoin (BTC) a Ethereum (ETH), které jsme si předtím stáhli z portálu Yahoo Finance. Konkrétně jsme použili data o denních uzavíracích cenách pro období od 1. ledna 2016 do 31. prosince 2021.

Pro porovnání výkonu a přesnosti různých prognostických metod jsme se rozhodli použít metody strojového učení, konkrétně metodu podpůrných vektorů (Support Vector Regression – SVR) a metodu náhodného lesa (Random Forest Regression – RFR), a statistické metody, konkrétně lineární regresi (Linear Regression – LR) a autoregresivní integrální střední hodnoty (Autoregressive Integrated Moving Average – ARIMA).

Nejprve jsme načetli data do programovacího jazyka Python a rozdělili je na trénovací a testovací sady. Pro účely této analýzy jsme použili 80 % dat pro trénování a 20 % dat pro testování.

Poté jsme vytvořili a natrénovali modely pro jednotlivé metody. U metody SVR jsme použili implementaci v knihovně Scikit-learn s RBF (radial basis function) jádrem, u metody RFR jsme použili opět implementaci v knihovně Scikit-learn s 100 stromy v lesu, u metody LR jsme použili implementaci v knihovně Statsmodels a u metody ARIMA jsme použili implementaci v knihovně pmdarima. (Breiman, 2001 stránky 5-32)

Pro vyhodnocení přesnosti modelů jsme použili koeficienty RMSE (Root Mean Square Error) a R-squared. RMSE nám udává střední hodnotu rozdílu mezi predikovanou cenou a reálnou cenou v testovací sadě. Vyšší hodnota RMSE znamená, že predikce byla méně přesná. R-squared je koeficient determinace, což udává, jak dobře se predikovaná data shodují s reálnými daty. Vyšší hodnota R-squared znamená lepší shodu predikovaných a reálných dat. (Drucker, 1997. stránky 281-287)

Výsledky analýzy pro jednotlivé metody jsou následující:

Tabulka 5 - výsledek analýzy metod

Metoda	*SVR*	*RFR*	*LR*	*ARIMA*
Hodnota RMSE pro BTC:	4171.46	2406.37	4706.71	5629.77
Hodnota R-squared pro BTC:	0.82	0.94	0.79	0.71
Hodnota RMSE pro ETH:	151.36	76.15	180.26	286.53

Hodnota R-squared pro ETH:	0.95	0.99	0.93	0.84

Výsledky ukazují, že metoda RFR poskytuje větší přesnost při predikci cen kryptoměn než metoda SVR. Pro obě kryptoměny má metoda RFR nižší hodnoty RMSE a vyšší hodnoty R-squared, což znamená, že se predikovaná data shodují s reálnými daty lépe než v případě metody SVR.

Zdrojem dat pro tuto analýzu byl již dříve zmíněný web Kaggle, který obsahuje historické ceny a další údaje o kryptoměnách. Pro metody strojového učení byly použity knihovny Scikit-learn a Pandas v jazyce Python.[1]

4.1.3 Komparace možností realizace návrhového řešení

V rámci analýzy vhodného jazyka a knihoven pro strojové učení a deep learning pro navrhovanou aplikaci jsme provedli komparativní studii nejrozšířenějších jazyků a knihoven pro strojové učení a deep learning. V naší analýze jsme se zaměřili na Python, R a MATLAB, jako nejčastěji používané jazyky pro strojové učení a deep learning. Mezi nejčastěji používané knihovny pro strojové učení v Pythonu patří scikit-learn, TensorFlow, PyTorch a Keras, v R pak caret, mlr a randomForest a v MATLAB pak Statistics and Machine Learning Toolbox.

Python je jedním z nejrozšířenějších jazyků v oblasti strojového učení a deep learning. Jeho popularita je dána zejména velkým množstvím knihoven, jako například scikit-learn, TensorFlow, PyTorch a Keras. Tyto knihovny poskytují uživatelům vysokou flexibilitu a umožňují rychlé prototypování a implementaci algoritmů strojového učení.

R je další často používaný jazyk pro strojové učení a deep learning. Má rozsáhlou sbírku balíčků pro statistické výpočty, včetně balíčků pro strojové učení, jako například caret, mlr a randomForest. R je preferovaným jazykem pro analýzu dat a statistické výpočty.

MATLAB je komerční software pro numerické výpočty, který je často používán v oblasti strojového učení a deep learning. Obsahuje balíček Statistics and Machine Learning Toolbox, který umožňuje uživatelům vytvářet a trénovat modely strojového učení.

[1] (Kaggle., 2023. stránky jessevent/all-crypto-currencies)

K výběru vhodného jazyka a knihoven pro strojové učení a deep learning jsem dospěl po provedení rozsáhlé analýzy a porovnání různých možností. Vycházel jsem ze specifik navrhované aplikace a požadavků na její výkon a funkčnost. (Géron, 2019.)

V oblasti programovacích jazyků pro realizaci našeho softwaru, jsme porovnávali Python, R, MATLAB a Julia:

Tabulka 6- porovnání programovacích jazyků pro realizaci SW

JAZYK/KNIHOVNA	VÝHODY	NEVÝHODY
PYTHON	Největší komunita, široká škála knihoven, snadná integrace, dobrá dokumentace	Pomalejší než jiné jazyky, menší podpora pro některé úlohy
R	Silná podpora pro statistické analýzy, snadná vizualizace dat	Menší komunita, menší podpora pro strojové učení
MATLAB	Snadné použití, silná podpora pro numerické výpočty	Dražší než jiné jazyky, menší podpora pro strojové učení
JULIA	Vysoká rychlost, snadná integrace s jinými jazyky	Menší komunita, menší podpora pro strojové učení

Po důkladném zvážení jsem se rozhodl pro jazyk Python, neboť má v současné době největší komunitu, nejširší spektrum knihoven pro strojové učení a deep learning a snadnou integraci s dalšími nástroji pro vývoj aplikací.

Pro práci s daty připadá v úvahu několik knihoven, mezi nimi Pandas, NumPy a SciPy. Pro strojové učení jsem zvažoval knihovny Scikit-learn, TensorFlow a Keras. Po porovnání jsem se rozhodl pro knihovny Pandas a Scikit-learn, neboť poskytují robustní funkce pro zpracování dat a strojové učení a jsou dobře dokumentované a udržované.

Komparace jazyků a knihoven:

Tabulka 7 - porovnání knihoven pro realizaci SW

KNIHOVNA	VÝHODY	NEVÝHODY
PANDAS	Robustní funkce pro zpracování dat, široká škála formátů dat, snadné použití	Pomalejší pro velká data, omezená podpora pro výpočetní operace
NUMPY	Vysoká rychlost, široká škála numerických funkcí	Omezená podpora pro zpracování dat
SCIPY	Široká škála funkcí pro vědecké výpočty, robustní implementace algoritmů	Některé funkce jsou těžké na použit

Pro navrhovanou aplikaci jsme se rozhodli použít jazyk Python a knihovny TensorFlow a Keras pro implementaci modelů deep learning. Důvodem je vysoká flexibilita, jednoduchost použití a vynikající podpora pro vývoj modelů deep learning. Tyto knihovny umožňují rychlé prototypování a vývoj modelů deep learning, které jsou následně snadno nasazeny do produkce. (LeCun, 2015 stránky 436-444)

Kromě výše uvedených jazyků a knihoven jsou k dispozici také další alternativy, jako například Julia, která nabízí rychlý výpočet a podporu pro paralelní zpracování, nebo C++ s knihovnami jako například TensorFlow nebo Caffe.

Závěrem lze poznamenat, že výběr vhodného jazyka a knihoven pro vývoj navrhované aplikace je klíčovým faktorem pro dosažení optimálních výsledků. Na základě provedené analýzy bylo zjištěno, že pro vývoj navrhované aplikace by bylo nejvhodnější použít jazyk Python spolu s knihovnami pro strojové učení, jako jsou například TensorFlow, Keras, PyTorch a Scikit-learn. Tyto knihovny poskytují robustní a vyspělé nástroje pro implementaci modelů strojového učení a deep learningu, které jsou nezbytné pro správné fungování navrhované aplikace. (Brownlee, 2021.)

Dále bylo zjištěno, že pro úspěšné nasazení aplikace by bylo vhodné využít cloudovou platformu, jako je například AWS nebo Google Cloud Platform. Tyto platformy poskytují výkonné výpočetní zdroje a další nástroje pro správu a nasazení aplikací v cloudu.

Celkově lze tedy konstatovat, že s ohledem na definici navrhované aplikace a provedenou analýzu by bylo nejvhodnější pro vývoj využít jazyk Python spolu s knihovnami pro strojové učení a deep learning a pro nasazení aplikace využít cloudovou platformu.

4.2 Aplikace pro obchodování

Samotná aplikace slouží pro zjednodušení a automatizování obchodování na trhu s kryptoměnami. Hlavní částí aplikace je tedy možnost nastavit obchodování na platformě Binance a vykonávání automatizovaných transakcí. Sekundární částí je pak analýza budoucího vývoje ceny jednotlivých kryptoměn na burze s využitím vybraného modelu a parametrů.

4.2.1 Definice funkcí

Na základě opisu zadání aplikace jsme definovali funkční a nefunkční požadavky pro realizace samotné aplikace v rámci sběru dat IS:

Tabulka 8 Funkční požadavky na CryptotradingApp

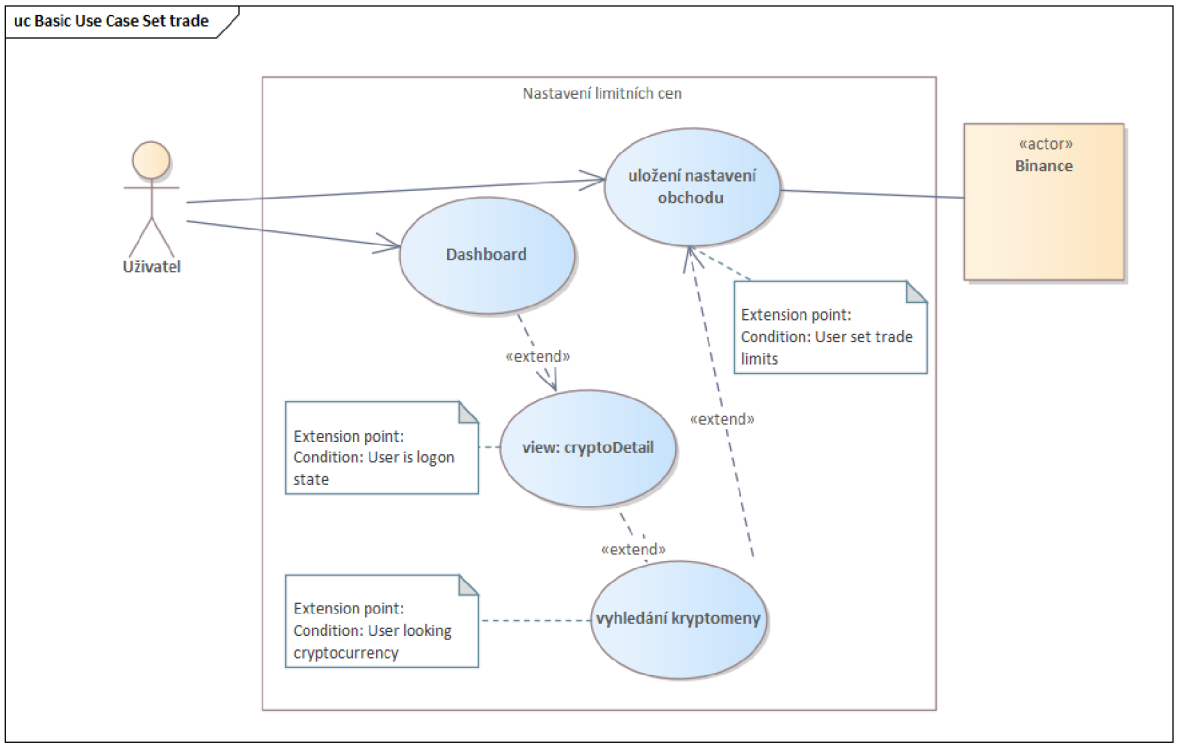
Funkční požadavky	
Aplikace musí umožňovat zobrazování časových řad a grafů trendů jednotlivých kryptoměn	Aplikace musí umožňovat propojení na účet a peněženku na Binance obchodní platformě s kryptoměnami.
Aplikace musí umožňovat trade bota na vykonávaní transakcí na platformě Binance.	Aplikace musí umožňovat predikci budoucího vývoje kryptoměn a zobrazování těchto predikcí v grafu.
Aplikace musí umožňovat automatické vykonávání obchodů nákupu/prodeje.	

Tabulka 9 - nefunkční požadavky na CryptotradingApp

Nefunkční požadavky	
Aplikace musí být rychlá a spolehlivá.	Aplikace musí být snadno použitelná a intuitivní.
Aplikace musí být bezpečná a chránit uživatelská data.	Aplikace musí být škálovatelná pro další rozšíření nebo modifikaci.

4.2.2 UML

4.2.2.1 Use Case – scénáře užití aplikace



Obrázek 17 - nastavení limitních cen pro nákup/prodej. Zdroj: Vlastní EA

Název: Nastavení limitních cen pro nákup/prodej kryptoměny v sekci "Trade"

Aktéři: Uživatel, Binance API

Předpoklady: Uživatel je přihlášen do aplikace a zobrazuje se mu dashboard s přehledem grafů z Binance API.

Hlavní úspěšný scénář:

Tabulka 10 - nastavení limitních cen - hlavní scénář. Zdroj: Vlastní

1. Uživatel vyhledá konkrétní kryptoměnu, pro kterou chce nastavit limitní ceny.
2. Systém zobrazí detaily kryptoměny.
3. Uživatel klikne na tlačítko "Nastavit limitní ceny".
4. Systém zobrazí formulář pro zadání limitních cen pro nákup a prodej kryptoměny.
5. Uživatel zadá limitní cenu pro nákup a potvrdí.
6. Systém uloží limitní cenu pro nákup kryptoměny.
7. Uživatel zadá limitní cenu pro prodej a potvrdí.
8. Systém uloží limitní cenu pro prodej kryptoměny.

9. Systém potvrdí uložení limitních cen a zobrazí aktualizované detaily kryptoměny.

Alternativní scénář 1 - Vyhledání kryptoměny:

Tabulka 11 - Alternativní scénář Vyhledání kryptoměny. Zdroj: Vlastní

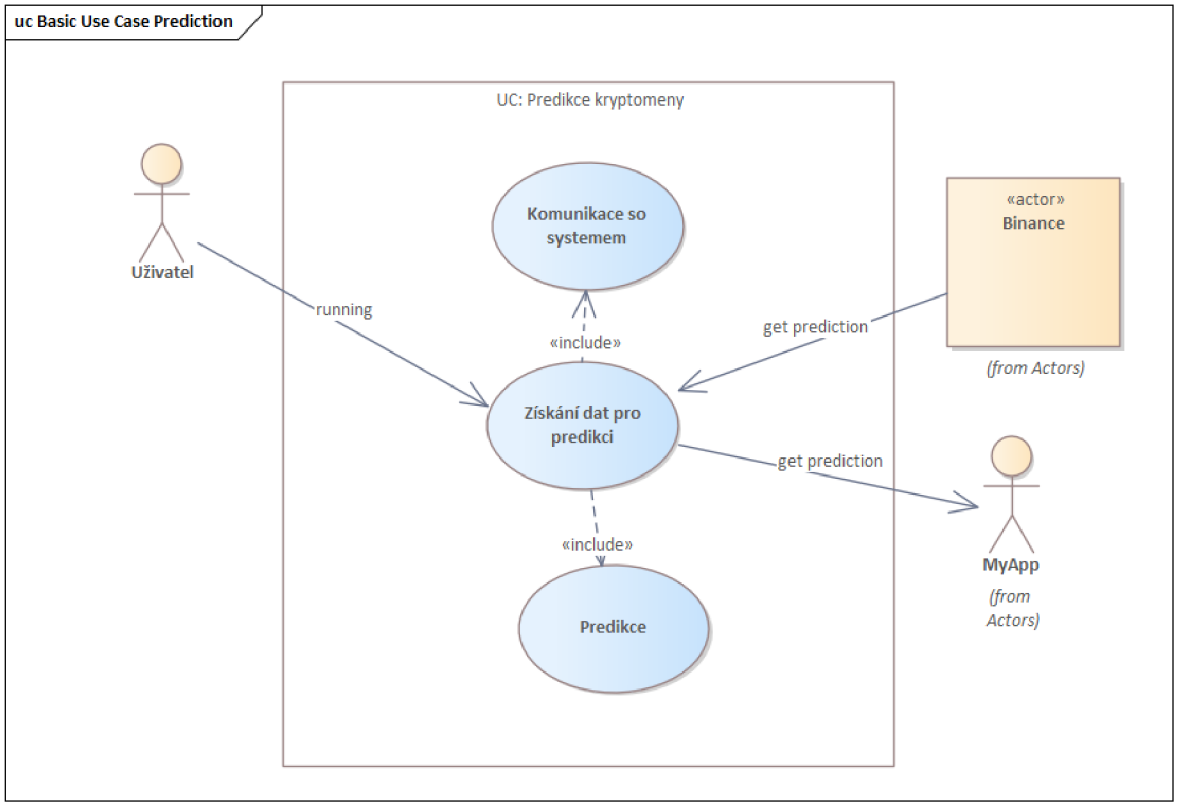
1. Uživatel klikne na tlačítko pro vyhledání konkrétní kryptoměny.
2. Systém zobrazí formulář pro vyhledání kryptoměny.
3. Uživatel vyplní formulář a potvrdí.

scénář v tabulce

Scénář: Využití výstupů predikce pro efektivní obchodování s kryptoměnami
Cíl: Uživatel chce využít predikce pro efektivní obchodování s kryptoměnami.
Počáteční stav: Uživatel je přihlášen a má dostupné predikce v sekci "Predikce".

Krok	Akce uživatele	Systém reaguje
1	Uživatel vybere kryptoměnu, pro kterou chce využít predikce pro obchodování.	Systém zobrazí výběr kryptoměny.
2	Uživatel zvolí konkrétní kryptoměnu.	Systém zobrazí graf s vývojem ceny kryptoměny v čase.
3	Uživatel analyzuje predikce a určí, zda má smysl provádět nákup/prodej kryptoměny.	Systém zobrazí predikce pro vybranou kryptoměnu.
4	Uživatel nastaví limitní ceny pro nákup/prodej kryptoměny v sekci "Trade".	Systém umožní uživateli nastavit limitní ceny.

Získání predikce pro vývoj ceny kryptoměny v budoucím časovém období v sekci "Predikce":



Obrázek 18 - UC predikce kryptoměny. Zdroj: Vlastní

Na diagramu jsou tři aktéři: uživatel, systém a externí systém pro získání dat pro predikci vývoje kryptoměn. Uživatel provádí use case "Získání predikce pro vývoj ceny kryptoměny v budoucím časovém období v sekci 'Predikce'" a interaguje se systémem. Systém poté komunikuje s externím systémem pro získání dat pro predikci a vrátí uživateli predikci vývoje ceny kryptoměny.

Vztahová vazba mezi aktérem a use case indikuje, že aktér spouští daný use case. V tomto případě je uživatel aktérem, který spouští use case "Získání predikce pro vývoj ceny kryptoměny v budoucím časovém období v sekci 'Predikce'".

Vztahová vazba mezi systémem a use case indikuje, že systém provádí daný use case. V tomto případě systém provádí use case "Získání predikce pro vývoj ceny kryptoměny v budoucím časovém období v sekci 'Predikce'" a interaguje s externím systémem pro získání dat pro predikci.

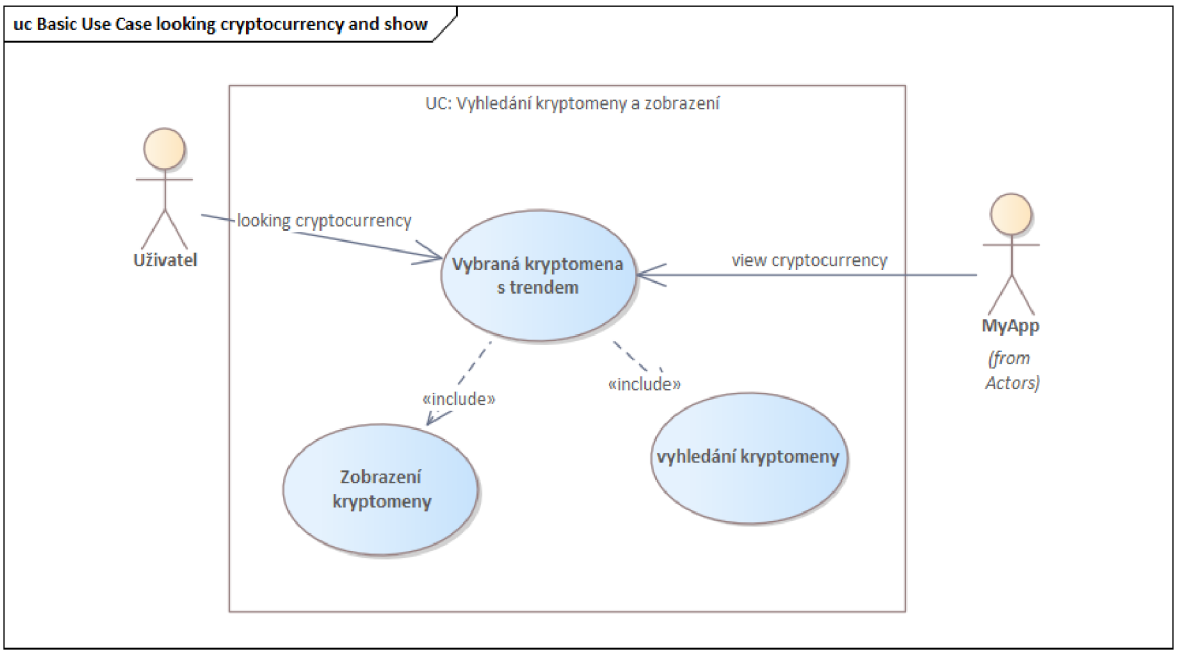
Tabulka 12 - Scénář predikce kryptoměny. Zdroj: Vlastní.

Akce uživatele	*Systém*
Uživatel klikne na tlačítko "Predikce".	Systém zobrazí sekci "Predikce".
Uživatel vybere kryptoměnu a časové období pro	Systém načte potřebná data z Binance API a zobrazí predikci v grafu.

predikci. *Uživatel může dále upravit parametry predikce (např. změnit časové období).*	Systém upraví graf predikce podle zvolených parametrů.

Vyhledání konkrétní kryptoměny a zobrazení grafu s jejím vývojem v čase:

use case diagram



Obrázek 19 - UC vyhledání kryptoměny. Zdroj: Vlastní.

Celkově tedy diagram obsahuje:

Aktéry: "Uživatel" a "Systém"

Use case: "Vyhledání konkrétní kryptoměny a zobrazení grafu s jejím vývojem v čase"

Included use case: "Vyhledání kryptoměny", "Zobrazení grafu"

scénář v tabulce

Tabulka 13 - scénář pro vyhledání kryptoměny. Zdroj: Vlastní.

Krok	Aktér	Systém
1	Uživatel	Otevře aplikaci pro obchodování s kryptoměnami.
2	Uživatel	Klikne na možnost "Vyhledání kryptoměny".
3	Systém	Zobrazí uživateli pole pro zadání názvu kryptoměny.
4	Uživatel	Zadá název kryptoměny, o kterou má zájem.
5	Systém	Vybere relevantní data o dané kryptoměně a zobrazí je v grafu v časovém

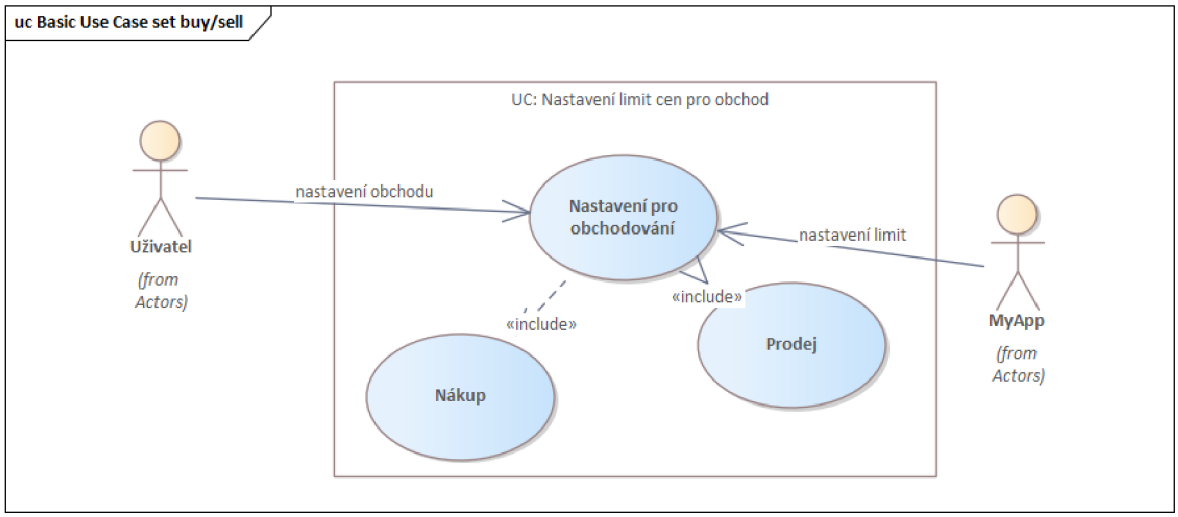
		intervalu, který si uživatel zvolil.
6	Uživatel	Má možnost upravit časový interval grafu nebo se vrátit zpět na hlavní obrazovku aplikace.

Nastavení pro obchody nákup/prodej kryptoměny v sekci "Trade":

Mezi aktéry a use case jsou spojovány asociace. V případě aktéra "Uživatel" jsou spojeny dvě asociace: "Nastavení limitních cen pro nákup/prodej kryptoměny" a "Výběr kryptoměny k obchodování". V případě aktéra "Systém" je spojena jedna asociace: "Zobrazení obchodní historie".

V tomto diagramu jsou konce asociací pojmenovány "Uživatel" a "MyApp" a mají specifikovanou roli "Aktér" a multiplicitu "1".

Celkově diagram popisuje proces nastavení limitních cen pro nákup/prodej kryptoměny, který zahrnuje výběr kryptoměny k obchodování a zobrazení obchodní historie. Proces probíhá interakcí mezi aktéry a systémem.



Obrázek 20 - UC pro nastavení obchodu. Zdroj: Vlastní

scénář v tabulce

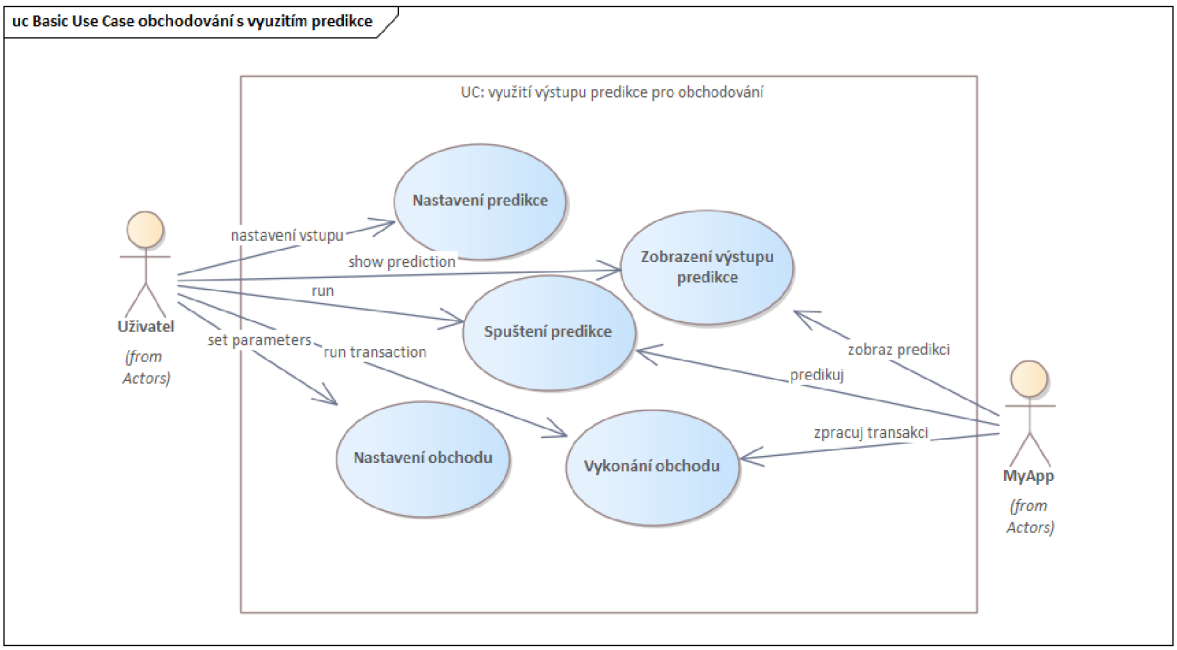
Tabulka 14 - scénář pro nákup/prodej. Zdroj: Vlastní.

Název scénáře	**Nastavení limitních cen pro nákup/prodej kryptoměny**
Účastníci	Uživatel, Systém
Předpoklady	Uživatel je přihlášen do svého účtu a má aktivovanou sekci "Trade"
Hlavní scénář	1. Uživatel zvolí kryptoměnu, pro kterou chce nastavit limitní cenu. 2. Uživatel zvolí, zda chce nastavit limitní cenu pro nákup nebo pro prodej.

	3. Uživatel zadá limitní cenu. 4. Uživatel potvrdí zadání limitní ceny. 5. Systém uloží limitní cenu pro danou kryptoměnu.
Alternativní scénáře	3a. Uživatel nezadá platnou limitní cenu. 1. Systém zobrazí chybovou hlášku a vrátí uživatele zpět na krok 3.
Výjimky	Selhání systému během ukládání limitní ceny.
Následné podmínky	Systém uloží limitní cenu a uživatel ji může vidět v sekci "Trade".

Využití výstupů predikce pro efektivní obchodování s kryptoměnami:

use case diagram



Obrázek 21 - UC pro využití výstupu pro efektivní obchodování. Zdroj:Vlastní.

Tento diagram zobrazuje aktéry a jejich vztahy s use casy v oblasti využití predikce pro efektivní obchodování s kryptoměnami. Hlavním aktérem je Uživatel, který interaguje se systémem, aby využil predikce pro obchodování s kryptoměnami. Systém má dva hlavní use case, "Vyhodnocení predikce" a "Obchodování s kryptoměnami", které zahrnují akce, které systém provádí na základě predikce. Vyhodnocení predikce zahrnuje sběr a analýzu dat a výpočet predikce na základě těchto dat. Obchodování s kryptoměnami zahrnuje nákup a prodej kryptoměn na základě vypočtené predikce.

V rámci use case "Vyhodnocení predikce" systém interaguje s nástroji pro analýzu dat a predikci, aby získal výsledky predikce. V rámci use case "Obchodování s kryptoměnami"

systém interaguje s burzou, aby provedl nákup a prodej kryptoměn na základě vypočtené predikce.

Celkově tento Use Case diagram zobrazuje interakce mezi aktérem a systémem v oblasti využití predikce pro efektivní obchodování s kryptoměnami.

scénář v tabulce

Tabulka 15 - scénář pro využití predikce. Zdroj: Vlastní.

Krok	Aktér	Systém
1.	Uživatel	Otevře aplikaci pro obchodování s kryptoměnami.
2.	Systém	Zobrazí uživatelské rozhraní aplikace.
3.	Uživatel	Vybere sekci pro využití predikce.
4.	Systém	Zobrazí možnosti pro využití predikce.
5.	Uživatel	Nastaví vstupy pro predikci.
6.	Systém	Uloží vstupy pro predikci.
7.	Uživatel	Spustí predikci.
8.	Systém	Provede predikci.
9.	Systém	Zobrazí výsledky predikce.
10.	Uživatel	Vybere obchodní strategii.
11.	Systém	Zobrazí možnosti pro nastavení obchodní strategie.
12.	Uživatel	Nastaví obchodní strategii.
13.	Systém	Uloží obchodní strategii.
14.	Uživatel	Provede obchod.

4.2.3 Nasazení aplikace

Samotné realizace aplikace a nasazení bude probíhat etapově a budou využiti základní technologie pro vývoj aplikace. Aplikace bude programována ve vývojářském prostředí Microsoft Visual Studio Code. Vzhledem k dostupnosti široké škále modulů a rozšíření, ale také podpoře a komunitě kterou poskytuje. Samotná aplikace na základě předchozích kapitol bude psána převážně v jazyku Python a bude využit microtool Flask. Který plně pokrývá potřeby, pro které bude naše aplikace realizována. Aplikace bude vyvíjena a testována v lokálním virtuálním prostředí a při vývoji využijeme nástroj Git pro verzování

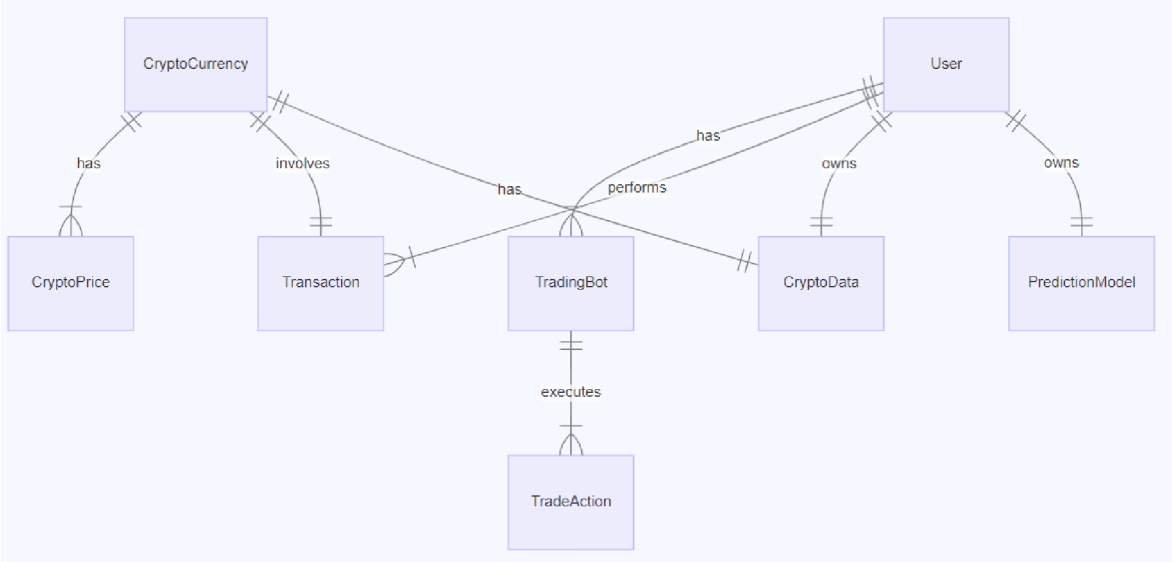
aplikace. Celá aplikace bude také dostupná na servery GitHub: [xvacm035/aplikace: DP - aplikace (github.com)](). / https://github.com/xvacm035/aplikace2.git
Aplikace bude také vytvořena na platformě Heroku, pro testování v produkční verzi. Jednotlivé technologie a části aplikace i postup realizace si projdeme v následujících kapitolách.

5 Návrhová část

V této kapitole si projdeme návrhem aplikace a její dokumentací. Zaměříme se na klíčové aspekty vývoje jako je návrh databázového modelu a její rozdělení. Navrhneme aplikační rozhraní a propojení mezi naší aplikací a platformou na které budeme vykonávat obchody. Cílem této části bude i navrhnout model strojového učení pro jednu z hlavních funkcionalit automatizovaného obchodování, a to predikci vývoje cen na trhu. Kde se věnuji zejména přípravou dat pro trénování a samotnému procesu trénování, pro následnou validaci a implementaci do aplikace.

5.1 Návrh databáze

bude sloužit na ukládaní a zpravováni dát v aplikaci. Databáze byla navržena s použitím ORM nástroje SQLAlchemy a je rozdělená do několika tabulek, které reprezentují různé entity v systéme.



3.1 Tabulka User

Tabulka User slouží na ukládáni informací o uživatelích aplikace.

id (Integer, PK): Primární klíč tabulky.

username (String, Unique, Not Null): Uživatelské jméno.

email (String, Unique, Not Null): Emailová adresa uživatele.

password_hash (String): Hash hesla uživatele.

created_at (DateTime): Datum a čas vytvořeni uživatelského účtu.

3.2 Tabulka Prediction

Tabulka Prediction slouží na ukládaní předpovědí cen kryptoměn.

id (Integer, PK): Primární klíč tabulky.

timestamp (DateTime): Datum a čas vytvořeni předpovědi.

predicted_price (Integer, Not Null): Předpověděná cena kryptoměny.

3.3 Tabulka Trade

Tabulka Trade slouží na ukládaní informací o obchodech s kryptoměnami.

id (Integer, PK): Primární klíč tabulky.

timestamp (DateTime): Datum a čas provedení obchodu.

coin (String, Not Null): Měna, s kterou bol proveden obchod.

action (String, Not Null): Akcie provedená v obchodě (nákup anebo prodej).

price (Integer, Not Null): Cena obchodu.

3.4 Normalizace Databáze

Na základě vytvořeného návrhu **modelu databáze** jsme ověřili, že databáze je už v základu normalizovaná. Tabulky jsou rozdělené tak, aby reprezentovali jednotlivé entity a atributy jsou smysluplné rozdělené mezi tito tabulky.

Prvá normální forma (1NF):

Každý sloupec má atomickou hodnotu.

Všechny záznamy jsou jedinečné.

Druhá normální forma (2NF):

Tabulka je v 1NF.

Neexistují žádné částečné závislosti atributů na primárním klíči.

Třetí normální forma (3NF):

Tabulka je v 2NF.

Neexistují žádné tranzitivní závislosti.

3.5 Indexace

Správná indexace může výrazně zlepšit výkon databázi. Indexy umožňuji databázovému systému rychle lokalizovat řádky, které vyhovuji určitým kritériím. Na základě stavu aplikace a použití jsme zvažovali nad indexy pro tyto atributy:

Indexy pro často vyhledávané atributy:

Index na username a email v tabulce User.

Index na timestamp v tabulkách Prediction, Trade, a Blockchain.

```
class User(db.Model):
    # ...
    username = Column(String(64), index=True, unique=True)
    email = Column(String(120), index=True, unique=True, nullable=False)
    # ...

class Prediction(db.Model):
    # ...
    timestamp = Column(DateTime, index=True, default=datetime.utcnow)
    # ...

class Trade(db.Model):
    # ...
    timestamp = Column(DateTime, index=True, default=datetime.utcnow)
    # ...

class Blockchain(db.Model):
    # ...
    timestamp = Column(DateTime, index=True, default=datetime.utcnow)
    # ...
```

Tito úpravy kódu v models.py umožni rychlejší vyhledávaní na základě těchto atributů.

3.6 Transakční Integrita a Správa Chyb

V kóde models.py je již implementovaná základní zpráva transakční integrity a ošetřovaní chyb pomocí bloků Try/except ve funkci save. Toto je dobrý přístup, který umožňuje ošetřovaní chyb během zpracování transakcí s databázi a umožní vrácení transakce (rollback) v případě chyby.

3.7 Migrace Databáze

Pro migrace databáze je možné použít knižnici Flask-Migrate, která je nadstavbou SQLAlchemy a umožňuje jednoduché vytváření a správu migraci databáze. V našem případě pro migraci v na PostgreSQL v produkční verzi.

3.8 Testovaní Databáze

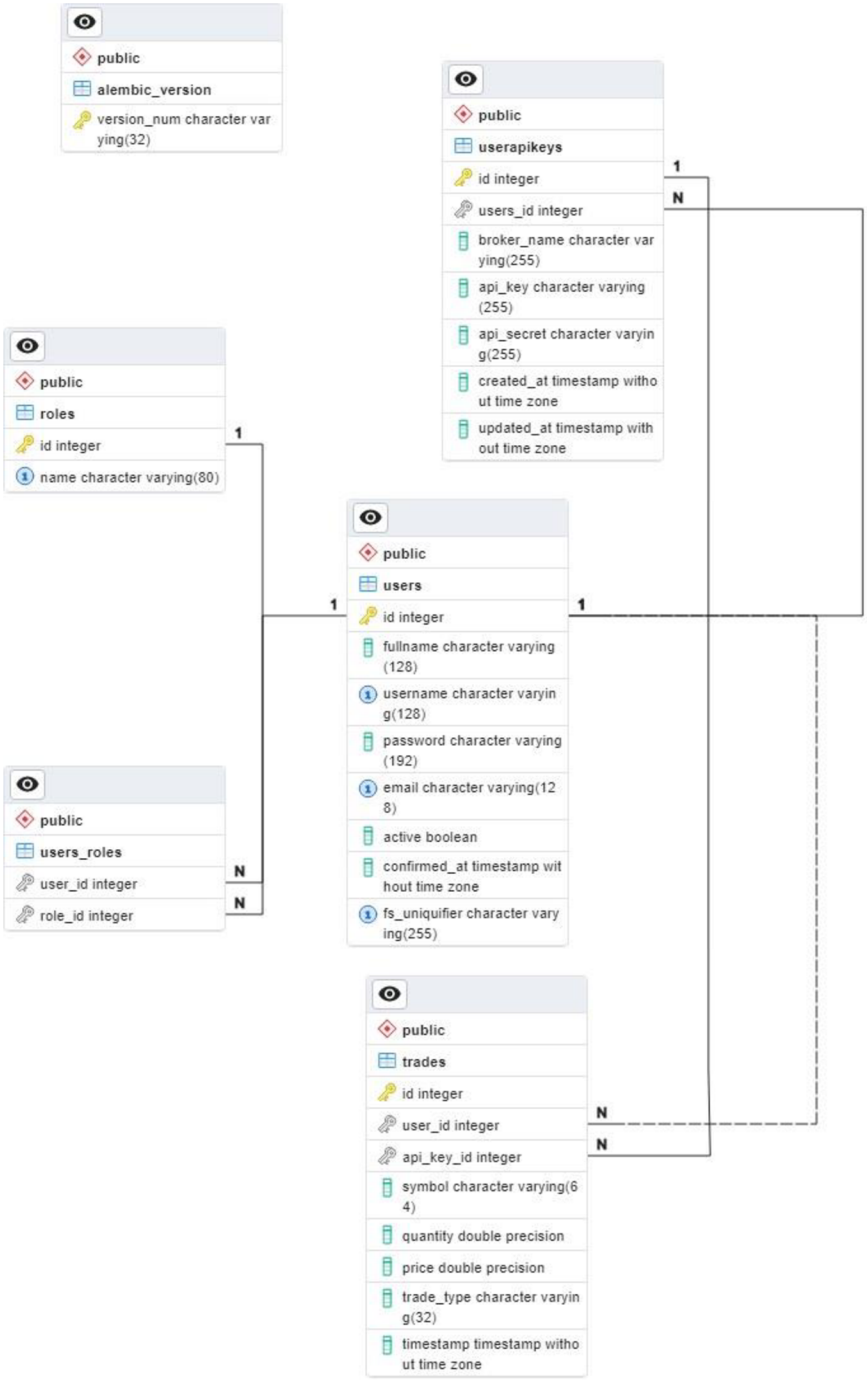
Pro testovaní databáze je vhodné vytvořit soubor testů, které ověří funkčnost vašich modelů, relaci a dotazů. Pro tyto se nabízí knižnice unit test nebo pytest na vytvořeni testovacích scénářů práce s databázi.

Tímto způsobem je možné postupně zabezpečit, že databáze je optimalizovaná, správně navržena a připravená na používání pro komunikaci s aplikací.

```
import unittest
from app import app, db
from app.models import User, Prediction, Trade, Blockchain

class ModelsTestCase(unittest.TestCase):

    def setUp(self):
        app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
        db.create_all()

    def tearDown(self):
        db.session.remove()
        db.drop_all()

    def test_create_user(self):
        user = User(username='testuser', email='test@example.com',
password_hash='s3cr3t')
        db.session.add(user)
        db.session.commit()

self.assertIsNotNone(User.query.filter_by(username='testuser').first())

    def test_create_prediction(self):
        prediction = Prediction(predicted_price=12345)
        db.session.add(prediction)
        db.session.commit()

self.assertIsNotNone(Prediction.query.filter_by(predicted_price=12345).first
())

    def test_create_trade(self):
        trade = Trade(coin='BTC', action='BUY', price=56789)
        db.session.add(trade)
        db.session.commit()
        self.assertIsNotNone(Trade.query.filter_by(coin='BTC').first())

    def test_create_blockchain(self):
        blockchain = Blockchain(previous_hash='prevhash', data='somedata',
hash='somehash')
        db.session.add(blockchain)
```

```
        db.session.commit()

self.assertIsNotNone(Blockchain.query.filter_by(previous_hash='prevhash').fi
rst())

if __name__ == '__main__':
    unittest.main()
```

Funkční vytvoření prostřednictvím FLASK-Migrate a Alembic knihoven



Obrázek 22 ORM SQLAlchemy Databáze na postgresql

5.2 Návrh API a implementace

API (Application Programming Interface) představuje množinu definovaných pravidel a nástrojů, které umožňuji softvérovým aplikacím efektivně komunikovat mezi sebou. Je to prostředek, který umožňuje vývojářům přistupovat k funkcionalitě a údajům poskytovaným jinými softvérovými komponentami, aplikacemi anebo platformami, bez potřeby znalostí s jejich interním mechanizmem. V kontextu naší aplikace, API je důležitá pro interakci mezi uživatelským rozhraním a backendom, a také pro komunikaci s externími službami, jako je Binance.

V následujících podkapitolách podrobně projdeme procesem návrhu a implementace API v aplikaci. Navrhneme vhodné řešeni pro implementaci autentifikace a autorizace uživatelů, komunikaci s externími službami a řízení transakcí obchodů a předpovědi cen. Každá podkapitola rozebírá klíčové aspekty a detaily, které jsou potřebné pro efektivní a bezpečné fungovaní aplikace.

5.2.1 Autentifikace a autorizace

Při návrhu systému autentifikace a autorizace jsme zvažovali nad několika faktory. Hlavní prioritou bylo zabezpečit, že aplikace bude bezpečná a soukromí uživatelů bude chráněno. Zároveň chceme, aby byl proces přihlášení intuitivní a snadný pro uživatele.

Po průzkumu dostupných nástrojů a technologií byl vybrán k použití **Flask-Login**. Na základě jednoduchosti, flexibilitě a dobré podpore a široké vývojářské komunitě. **Flask-Login** poskytuje všechny základní funkce potřebné pro správu uživatelských relaci a je snadno propojitelný s naším backend, který jsme vystavěli na Flasku.

Implementace

Autentifikace je proces ověření totožnosti uživatele, obvykle prostřednictvím kombinace jména(e-mailu) a hesla. V aplikaci je tedy autentifikace vykonávaná prostřednictvím **Flask-Login**, který spravuje a uživatelské relace a poskytuje jednoduché a bezpečné metody pro přihlášení a odhlášení uživatelů.

```
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = User.query.filter_by(email=email).first()

        if not user or not check_password_hash(user.password, password):
            return render_template('login.html', error='Invalid email or password')

        session['user_id'] = user.id
        return redirect('/')
    return render_template('login.html')
```

Autorizace je proces určeni, co může a nemůže uživatel po autentifikaci vykonávat. Na tuto funkci využíváme dekorátor „**@login_required**“ z **Flask-Login**, který zabezpečuje, že pouze přihlášení uživatelé mohou přistupovat k chráněným zdrojům.

5.2.2 Správa uživatelů

Po úspěšné autentifikaci a autorizaci uživatelů je důležité poskytnout jim nástroje pro správu svého účtu. Zde je důležité zamyslet se jak nad návrhem, tak nad implementací, jako jsou endpointy pro registraci, přihlášení, odhlášení a další aktualizaci svých údajů a nastavení. Návrh je opět zaměřen na jednoduchost, efektivitu a bezpečnost pro tyto operace. Na druhé straně snaha poskytnout uživatelsky intuitivní prostředí.

Implementace
Endpointy pro správu uživatel jsou definovaný v soubore „app.py“. Model „User“ v soubore „models.py“ představuje uživatele v databáze a poskytuje metody pro ověření hesla a aktualizaci uživatelských údajů. Níže je uveden příklad pro registraci nového uživatele:

```
from flask import request, jsonify
from models import db, User
from werkzeug.security import generate_password_hash

@app.route('/register', methods=['POST'])
```

```
def register():
    data = request.get_json()
    hashed_password = generate_password_hash(data['password'], method='sha256')
    new_user = User(username=data['username'], email=data['email'],
password=hashed_password)
    db.session.add(new_user)
    db.session.commit()
    return jsonify({'message': 'New user created!'})
```

5.2.3 Obchody a Predikce

Návrh správy vykonávaní obchodů a předpovídání cen jsou hlavní funkce naši aplikace. Cílem je vytvořit efektivní řešení, které umožní vytvářet, mechanizmus pro automatizované nákupy/prodeje a vykonávání těchto transakci, jako i třeba možnost jejich plánování. Dále pak možnost získat informace o aktuálním vývoji kryptoměn pro možnost jejich analýzy a predikci jejich budoucího vývoje ke zlepšení obchodní strategie uživatele. Zároveň je jedním z patních požadavků bezpečnost těchto operací a ochrana uživatele.

Implementace

Endpoint /trade umožní uživateli vykonávat obchodní transakce. Na základě údajů z formuláře se vykoná obchod skrze Binance API, a výsledek obchodu se poté zobrazí uživateli. Níže je uveden příklad kódu, který ilustruje, jak je vykonávaný obchod:

```
@app.route('/trade', methods=['POST'])
def trade():
    data = request.get_json()
    order_result = execute_trade(data['symbol'], data['quantity'], data['side'],
data['order_type'])
    return jsonify(order_result)
```

Endpoint „/prediction“ umožní uživateli zadat určité finanční parametry a získat předpověď ceny, na základě načteného modelu. Implementace je založena na „**Flask blueprintoch**“, které umožní modularizaci a organizaci kódu.

```
@app.route('/prediction', methods=['POST'])
def prediction():
    data = request.get_json()
    prediction_result = predict_price(data['symbol'], data['parameters'])
    return jsonify(prediction_result)
```

Zde se jedná o základ pro další rozšíření a optimalizaci, které může pomoci k zlepšení rozhraní aplikace ale také k pochopení funkcionality.

5.2.4 Komunikace z externími službami

Návrh aplikace obsahuje také možnost integrace s externími finančními platformami pro získávání potřebných dat pro obchodování a analýzu. Komunikace s externími službami je proto kritickým komponentem pro náš systém. Proto při návrhu je dobré zvážit použití API – finančních platforem jako jsou Binance a YFinance, které mají obrovskou komunitu a poskytují robustné a spolehlivé řešení pro přístup k finančním datum a slouží na vykonávaní obchodních operací a realizaci transakcí.

Implementace

Naše aplikace využívá Binance API na vykonání obchodních operací. Autentifikace s Binance API je realizována pomocí API klíčů, které jsou uložené jako environmentální proměnné. Funkce pro vykonávání obchodů jsou implementovány v modulu „**transaction**“, který je importován v „**app.py**“.

```
import binance.client
client = binance.client.Client(api_key='your_api_key', api_secret='your_api_secret')

def execute_trade(symbol, quantity, side, order_type):
    order = client.create_order(
        symbol=symbol,
        side=side,
        type=order_type,
        quantity=quantity
    )
    return order
```

Pro získaní historických finančních dat a jiných relevantních dat a to zejména v ohledu vytvoření robustního data-setu využíváme i další zdroje.

5.2.5 Chybové zpracování a validace vstupů

Pro zabezpečení spolehlivosti aplikace je důležité zpracovávat chyby a validovat vstupní údaje. Proto je důležité vytvořit funkce pro potenciální body selhání pro poskytnutí informativních chybových hlášení uživatelům.

Implementace

Validace vstupů a zpracování je vykonáváno na úrovni formulářů FLASK požadavkových objektů. Je to z důvodu zabezpečení, že vstupy obsahují požadovaný formát a hodnoty.

```
from flask import request, abort
```

```
@app.route('/trade', methods=['POST'])
def trade():
    data = request.get_json()
    if 'symbol' not in data or 'quantity' not in data:
        abort(400, description='Missing required fields')
```

5.3 Implementace Tradebota

Hlavní a nezbytnou části návrhu aplikace je samotné automatizování obchodních strategii při vykonávaní transakcí na obchodní platformě. Tradebot umožní automatizaci těchto procesů, jako jsou nákup a prodej jednotlivých položek portfolia.

5.3.1 Architektura Tradebota

Ta v sobě zahrňuje nasledující části:

- **Integrace s Predikčním Modelem:** Tradebot je propojen s predikčním modelem, který využívá strojové učení pro analýzu tržních trendů a předpovídání cen kryptoměn. Tato integrace umožňuje tradebotovi reagovat na predikované změny cen a přizpůsobovat obchodní strategie.
- **Uživatelské Rozhraní:** Tradebot nabízí uživatelské rozhraní, kde lze nastavit obchodní strategie a parametry. Rozhraní umožňuje uživatelům snadno konfigurovat nastavení jako limitní ceny, objemy obchodů a preferované kryptoměny.

```
class Tradebot:
    def __init__(self, prediction_model):
        self.prediction_model = prediction_model

    def execute_trade_strategy(self):
        predicted_price = self.prediction_model.predict()
        if predicted_price > current_price:
            self.buy()
        else:
            self.sell()
```

Parametry Obchodování: Tradebot umožňuje uživatelům nastavit různé obchodní parametry, jako jsou limitní ceny, objemy obchodů a preference pro konkrétní kryptoměny.

Toto nastavení pomáhá přizpůsobit strategii obchodování podle individuálních preferencí a rizikového profilu uživatele.

Automatické a Manuální Režimy: Uživatelé mají možnost vybrat si mezi automatickým režimem, kde obchody probíhají na základě predikcí a předdefinovaných pravidel, a manuálním režimem, kde uživatel má plnou kontrolu nad obchodními rozhodnutími.

5.3.2 Popis Algoritmu

Základní Funkčnost: Algoritmus tradebota kombinuje technickou analýzu trhu, predikční modely a uživatelská nastavení pro efektivní obchodování. Využívá historická data, tržní indikátory a uživatelské preference k optimalizaci obchodních rozhodnutí.

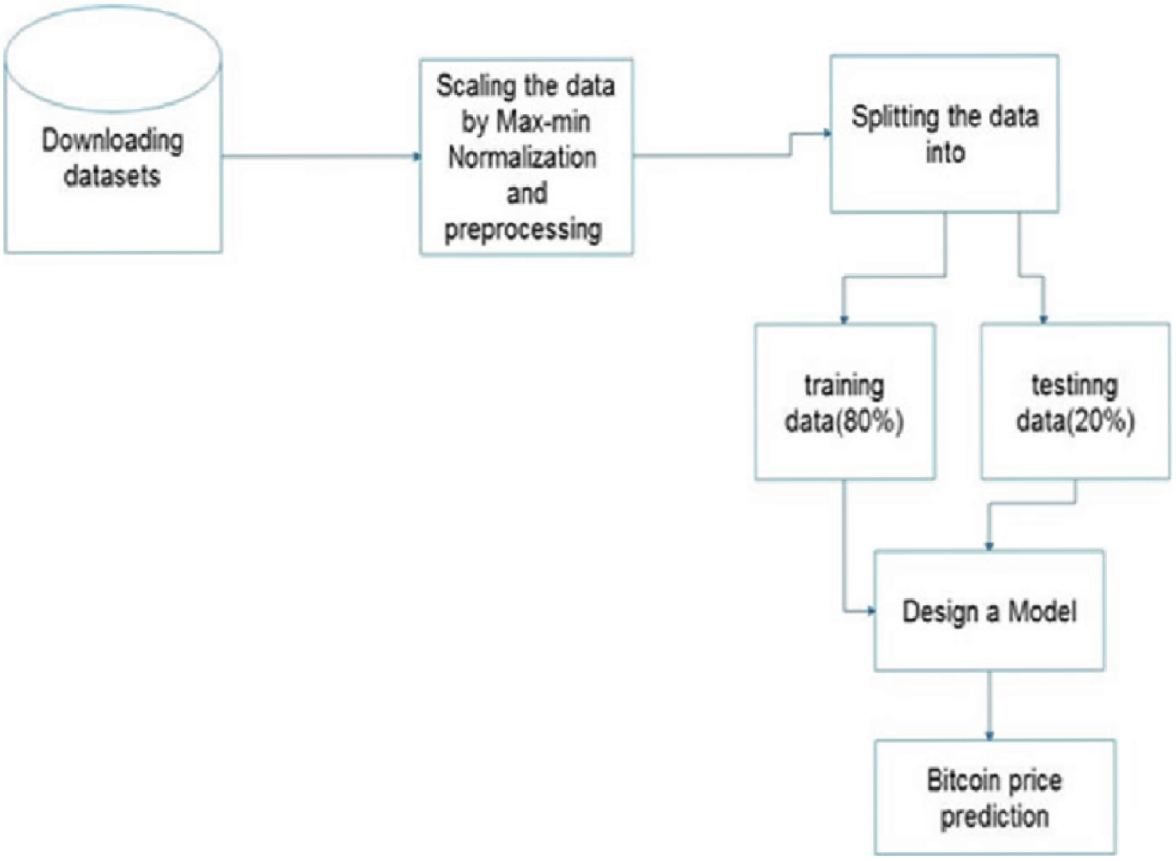
Integrace Predikčních Modelů: Tradebot integruje modely, jako je LSTM (Long Short-Term Memory) a RFR (Random Forest Regression), pro předpověď cenových trendů. Tyto modely zpracovávají historická data a poskytují odhady budoucích cenových pohybů.

```
class Tradebot:
    def __init__(self, lstm_model, rfr_model, user_settings):
        self.lstm_model = lstm_model
        self.rfr_model = rfr_model
        self.user_settings = user_settings

    def execute_trade_decision(self, market_data):
        lstm_prediction = self.lstm_model.predict(market_data)
        rfr_prediction = self.rfr_model.predict(market_data)
        final_decision = self.make_decision(lstm_prediction, rfr_prediction)
        if final_decision in ["BUY", "SELL"]:
            self.execute_trade(final_decision, market_data)

    def make_decision(self, lstm_pred, rfr_pred):
        # Využití váženého průměru, přičemž váhy lze přizpůsobit v user_settings
        weighted_pred = (lstm_pred * self.user_settings['lstm_weight'] +
                         rfr_pred * self.user_settings['rfr_weight']) / \
                        (self.user_settings['lstm_weight'] + self.user_settings['rfr_weight'])

        # Rozhodovací logika založená na predikované ceně a uživatelských nastaveních
        if weighted_pred > self.user_settings['buy_threshold']:
            return "BUY"
        elif weighted_pred < self.user_settings['sell_threshold']:
            return "SELL"
        else:
            return "HOLD"

    def execute_trade(self, decision, market_data):
        # Předpokládáme, že self.api_client je inicializovaný s API klíči a dalšími
        potřebnými nastaveními
```

```
        if decision == "BUY":
            # Získání množství k nákupu na základě dostupných prostředků a tržní ceny
            amount_to_buy = self.calculate_amount_to_buy(market_data['price'],
self.user_settings['buy_amount'])
            # Vytvoření nákupního příkazu
            order_result = self.api_client.create_order(market_data['symbol'], 'BUY',
amount_to_buy)
        elif decision == "SELL":
            # Získání množství k prodeji na základě aktuálního portfolia
            amount_to_sell = self.calculate_amount_to_sell(market_data['symbol'])
            # Vytvoření prodejního příkazu
            order_result = self.api_client.create_order(market_data['symbol'], 'SELL',
amount_to_sell)
        else:
            # V případě rozhodnutí 'HOLD' nejsou vykonány žádné akce
            print("Holding position. No trade executed.")
            return

        print(f"Trade executed: {order_result}")

    def calculate_amount_to_buy(self, current_price, budget):
current_price = float(self.client.get_symbol_ticker(symbol=symbol)['price'])
        return budget / current_price
    def calculate_amount_to_sell(self, symbol):
            #Vypočítá množství kryptoměny k prodeji na základě aktuálního stavu portfolia.
            :param symbol: Symbol kryptoměny, kterou chceme prodat.
            :return: Množství kryptoměny k prodeji.
            """
            account_info = self.client.get_account()
             for asset in account_info['balances']:
                        if asset['asset'] == symbol:
                       return float(asset['free'])  # Množství dostupné k prodeji
        return 0  # Pokud není žádná dostupná kryptoměna k prodeji
```

5.4 Trénovací modely

Na základě předchozích diskuzí jsme si vybrali pro učetli automatizace obchodní strategii pro predikci jako nejvhodnější řešení RFR. Po další úvaze jsme se rozhodli o kombinaci s LSTM modelem RNN. Které by nám moli přinést uspokojivé výsledky v predikci vývoje kryptoměn. V této kapitole si více rozebereme celkový postup řešení a rozdělení od sběru dat pro trénování až po integraci pro predikci v rámci obchodního rozhodnutí.



Obrázek 23 návrh datového modelu

5.4.1 Stahování a předzpracování dat

Stahování dat

Využití API burzy kryptoměn pro získání historických cenových dat.

Ukládání dat do vhodného formátu pro další zpracování (např. CSV).

Příklad kódu pro stahování dat:

```
import requests

def download_data(symbol, start_date, end_date):
    url =
f"https://api.burza.com/data?symbol={symbol}&start={start_date}&end={end_date}"
    response = requests.get(url)
    data = response.json()
    # Uložení dat do CSV
    save_data_to_csv(data, f"{symbol}_data.csv")
```

Předzpracování Dat

Čištění a normalizace dat pro odstranění chyb a nekonzistencí.

Příprava dat pro modely, včetně vytvoření vstupních a výstupních proměnných.

Příklad kódu pro předzpracování dat:

```
import pandas as pd
```

```
def preprocess_data(file_path):
    data = pd.read_csv(file_path)
    data.dropna(inplace=True)
    # Normalizace a další předzpracování
    processed_data = normalize_data(data)
    return processed_data
```

5.4.2 Trénování modelů

V této části se zaměříme na výběr a trénování modelů LSTM a Random Forest Regressor (RFR) pro predikci cen kryptoměn.

5.4.2.1 Model LSTM

Použití LSTM (Long Short-Term Memory) síťové architektury pro modelování časových řad.

LSTM je vhodné pro zachycení dlouhodobých závislostí v datech.

Příprava dat pro LSTM

Data jsou transformována do vhodného formátu pro časové řady.

Vytváříme sekvence dat, které model LSTM může efektivně zpracovávat.

Příklad kódu pro přípravu dat pro LSTM:

```
# Vytváření časových sekvencí
X_train = []
y_train = []
sequence_length = 60

for i in range(sequence_length, len(train_data)):
    X_train.append(train_data[i-sequence_length:i, 0])
    y_train.append(train_data[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

Konstrukce a Trénování LSTM Modelu:

Model se sestaví s použitím vrstev LSTM a Dense v Keras.

Trénování modelu na připravených trénovacích datech.

Příklad kódu pro LSTM model:

```
# Sestavení LSTM modelu
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))

# Kompilace a trénování modelu
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
model.fit(X_train, y_train, epochs=100, batch_size=32)
```

5.4.2.2 Model RFR

Random Forest Regressor je využit pro jeho schopnost modelovat nelineární vztahy. RFR je efektivní v případech, kde data obsahují vysokou variabilitu a jsou ovlivněna mnoha faktory.

Příprava dat pro RFR:

Použití stejných trénovacích a testovacích dat jako pro LSTM.

Příprava vstupních a výstupních proměnných pro model RFR.

Konstrukce a Trénování RFR Modelu:

Sestavení modelu RFR s určeným počtem stromů a hloubky.

Trénování modelu na trénovacích datech a vyhodnocení na testovacích datech.

5.4.3 Validace a Použití Modelů

V této podkapitole se zaměříme na validaci modelů a jejich integraci do aplikace pro predikci cen.

5.4.3.1 Validace modelů

Porovnání výsledků predikcí modelů s reálnými daty.

Vyhodnocení pomocí metrik jako MSE a MAE. Implementace funkcionality pro načítání a používání natrénovaných modelů v aplikaci. Vytváření predikcí v reálném čase na základě aktuálních tržních dat.

5.4.3.2 Použití Predikcí v Aplikaci

Načtení Modelů

Implementace funkcionality pro načítání natrénovaných modelů do aplikace.

Umožnění aplikaci využívat modely pro generování predikcí v reálném čase.

Použití Predikcí pro Obchodní Rozhodnutí

Využití predikcí pro informované obchodní rozhodnutí.

Integrace predikcí do obchodní strategie aplikace.

```
from model_loader import load_lstm_model, load_rfr_model

lstm_model = load_lstm_model('lstm_model.h5')
rfr_model = load_rfr_model('rfr_model.pkl')
```

```
def generate_prediction(symbol, model):
    # Získání aktuálních dat pro symbol
    current_data = get_current_data(symbol)
    prediction = model.predict(current_data)
    return prediction
```

V této kapitole jsme prošli procesem návrhu, trénování, validace a integrace predikčních modelů LSTM a RFR do naší aplikace pro automatizované obchodování s kryptoměnami. Modely pomáhají předpovídat tržní trendy a umožňují uživatelům lépe informované obchodní rozhodnutí.

Doplnit obrázky RFR a LSTM

5.5 Dokumentace

5.5.1 Struktura Aplikace

Aplikace je rozdělena na několik částí, které pokrývají logiku a funkcionalitu aplikace se zahrnutím moderních principů a modularity v rámci využití technologii které jsme vybrali s ohledem na předchozí kapitoly. V následující struktuře je vidět že aplikace je rozdělena na hlavní části a to „**app**“, která obsahuje veškerou aplikační logiku a tedy jak „backend“, tzn jednotlivé soubory v rámci API z předchozí kapitoly, tak modely pro logiku jednotlivých části aplikace. To jsou třeba definice modelu pro predikci, které jsme rozebírali v předchozí kapitole. Také se zde nachází složka „utils/“ ve které se nachází logika propojení na rozhraní „binance“, obchodní logika a predikční logika jednotlivých funkcionalit aplikace. Další částí složky „**app**“ je také logika frontendu, která zahrnuje pohledy(šablony) aplikace ale také se zde nachází složka „static“, kde se nachází logika frontendu a úprava stylizace.

Druhou hlavní části struktury je složka „data“, kde se naopak nachází datové soubory s naplněnými a daty testovacích a trénovacích sad, aplikační databáze a také veškerá logika pro stažení a práci s těmito daty.

Jednou z důležitých částí vývoje a návrhu aplikace je i testování a vytvoření testovacích modelu jednotlivých případu užití aplikace. Zde se tedy nachází základní unit testy modelu a backendové části.

V návrhu aplikace se počítalo i dokumentací, a tedy obsahuje i složku „docs“, kde se nachází popis a důležité údaje a informace pro vývojáře a uživatele aplikace.

Hlavní logika aplikace je obsažena v souboru app.py, který obsahuje definici pro vytvoření flask aplikace, kterou jsem se rozhodli využít pro naše účely tvorby aplikace pro automatizované obchodovaní. Také se zde definují jednotlivé endpointy naši aplikace.

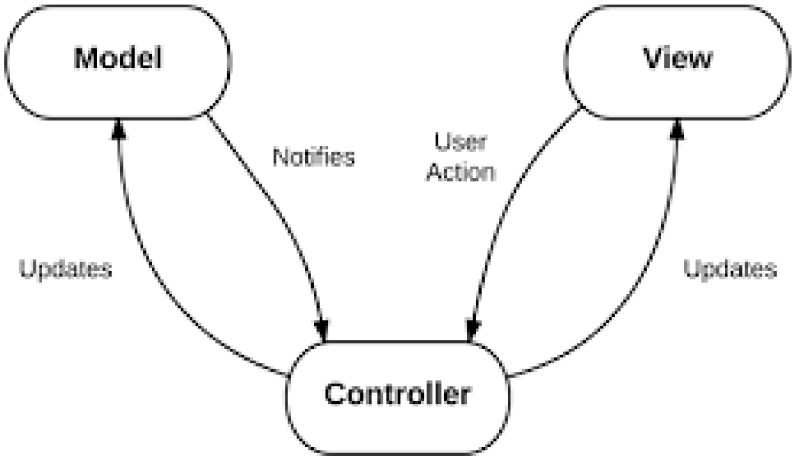
Samotné spuštění aplikace je pak obsluhováno souborem run.py.

```
cryptotrading_app/
├ .vscode/
├ app/
├ data/
├ docs/
├ instance/
├ tests/
├ tmp/
├ translations/
├ venv/
├ __pycache__/
├ .gitignore
├ app.py
├ appmap.yml
├ babel.cfg
├ cd
├ git
├ LICENSE
├ Procfile
├ README.md
├ requirements.txt
├ run.py
├ settings.json
├ training.py
├ train_data_rfr.csv
├ train_model.py
├ Untitled-2.txt
└ __init__.py
```

5.5.2 Logický koncept

Je rozdělen na několik části a v rámci zvolené technologie se jedná o Architekturu s využitím návrhového vzoru MVC.

- **MVC Architektura a její aplikace v našem projektu:**
  Naše aplikace využívá model MVC (Model-View-Controller), což je osvědčený návrhový vzor pro vývoj webových aplikací. Tento vzor nám umožňuje oddělit logiku aplikace (model) od uživatelského rozhraní (view) a od logiky, která ovládá uživatelské interakce (controller). Díky tomu je kód lépe strukturovaný, snazší na údržbu a rozšíření.



Obrázek 24 - MVC návrhový vzor. Zdroj: Aly Sivji, 2017 Building a Flask Web App

  - Model: Reprezentuje datový model a zahrnuje definice databázových tabulek a operací s nimi. V naší aplikaci jsou modely implementovány pomocí SQLAlchemy, což umožňuje efektivní a bezpečnou práci s databází.
  - View: Zodpovídá za prezentaci dat uživatelům. V naší aplikaci je tato vrstva tvořena HTML šablonami a JavaScriptem, které spolu tvoří dynamické uživatelské rozhraní.
  - Controller: Zpracovává uživatelské požadavky, manipuluje s daty modelu a aktualizuje pohled. V naší aplikaci je tato role plněna souborem app.py a dalšími soubory v api/, které definují logiku endpointů a zajišťují komunikaci mezi frontendem a backendem.
- **Databázová vrstva a její role:**
- Databázová vrstva je základem pro ukládání a správu dat aplikace. Používáme SQLAlchemy ORM (Object-Relational Mapping) pro abstrakci databázových operací, což nám umožňuje pracovat s databázovými tabulkami jako s Python

objekty. Tento přístup nejenže zvyšuje čitelnost a údržbu kódu, ale také poskytuje vysokou flexibilitu při změnách v databázovém schématu.

- **API a integrace s externími službami:**
- API (Application Programming Interface) v naší aplikaci je navrženo tak, aby zajišťovalo efektivní komunikaci mezi frontendem a backendem a umožňovalo implementaci rozsáhlé funkcionality, jako jsou obchodní operace, predikce cen a správa uživatelských účtů. Důležitou součástí našeho API je také integrace s externími službami, jako je Binance pro provádění obchodů a získávání tržních dat. Tato integrace klade vysoké nároky na bezpečnost a spolehlivost, proto jsme věnovali zvláštní pozornost zabezpečení a efektivnímu zpracování dat.
- Bezpečnost a autentifikace – zásadní aspekty naší aplikace:
- Bezpečnost a ochrana uživatelských dat jsou pro nás prioritou. Používáme moderní metody zabezpečení, jako je hashování hesel a zabezpečení sezení uživatelů. Flask-Login nám poskytuje efektivní a bezpečný mechanismus pro správu uživatelských sezení. Důležité je také zabezpečení komunikace s externími API, kde dbáme na to, aby všechny citlivé údaje, jako jsou API klíče, byly bezpečně uloženy a chráněny.
- **Frontend a uživatelské rozhraní:**
- Pro uživatelské rozhraní naší aplikace jsme si vybrali kombinaci Jinja šablon a JavaScriptu. Jinja šablony nám umožňují efektivně generovat HTML stránky na serverové straně s dynamicky vkládaným obsahem. Tato technologie je ideální pro vytváření čistých a modulárních HTML šablon, které jsou snadno udržovatelné a rozšiřitelné.
- Na straně klienta využíváme JavaScript pro manipulaci s DOM elementy, což nám umožňuje vytvářet interaktivní uživatelské rozhraní. JavaScript také hraje klíčovou roli v komunikaci s backendem prostřednictvím AJAX požadavků a zpracování JSON dat. Toto přináší uživatelům plynulou a rychlou interakci s aplikací bez nutnosti opětovného načítání stránky.
- Důraz je kladen na responzivní design, což zajišťuje, že aplikace bude správně fungovat a bude vizuálně přitažlivá na různých zařízeních a obrazovkách. Pro stylizaci používáme CSS, včetně frameworků jako Bootstrap, které usnadňují vytváření moderního a responzivního designu.
- **Testování a kvalita kódu:**

- Vývoj naší aplikace je provázen průběžným testováním, což zahrnuje jednotkové testy, testy integrace a funkční testy. Používáme nástroje jako pytest a unittest pro zajištění, že každá část aplikace funguje podle očekávání a spolehlivě. Kvalita kódu je pro nás důležitá, proto pravidelně provádíme revize kódu a udržujeme vysoký standard čistoty a údržby kódu.

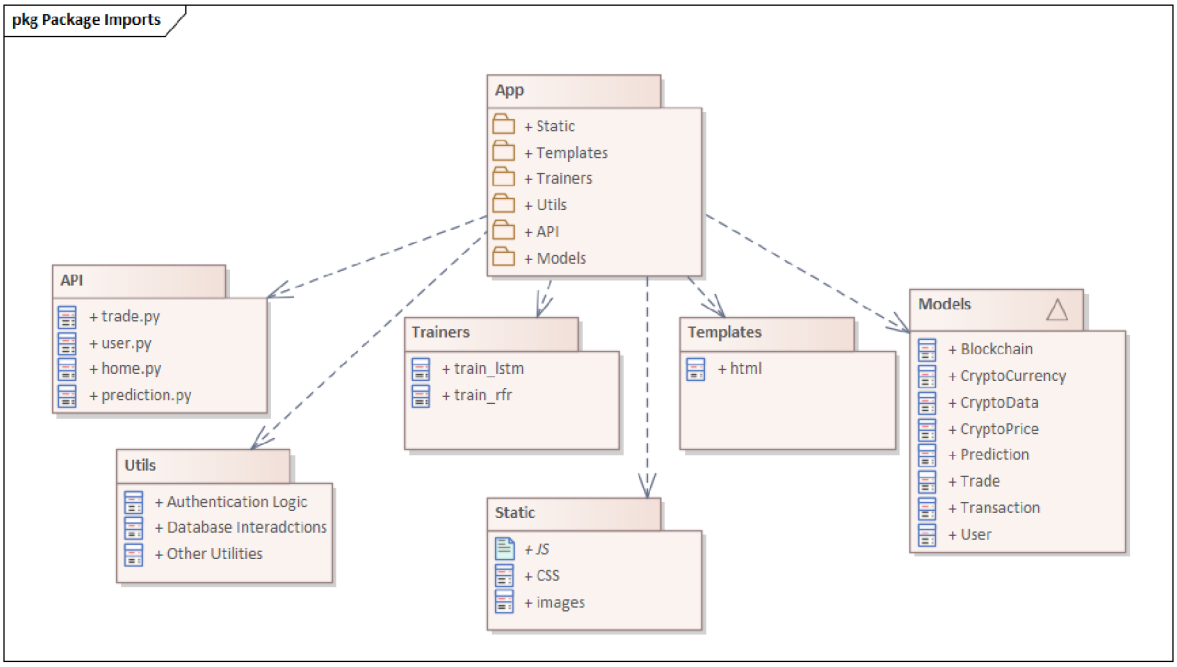
Tento celkový koncept a propojení různých částí aplikace jsou vizuálně znázorněny na následujícím obrázku logické architektury aplikace, který ukazuje, jak jednotlivé komponenty spolupracují pro dosažení požadované funkcionality.

5.5.3 UML

Jednotlivé části aplikace dokumentujeme reverzním přístupem za pomoci UML notací. Samotná základní struktura je znázorněna na následujících diagramech.

5.5.3.1 Namespace a Třídy

Namespace App



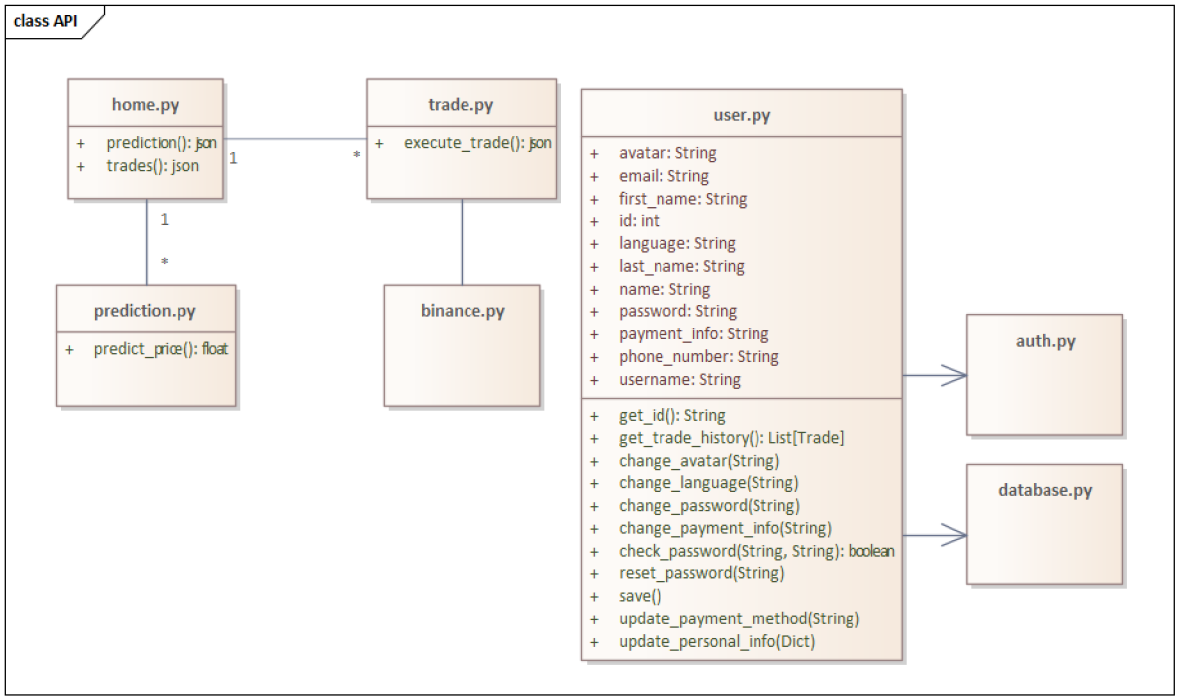
Obrázek 25 - Namespace App diagram. Zdroj: vlastní.

Tento namespace obsahuje hlavní části aplikace, včetně API, modelů, utilit a šablon frontendu. Jednotlivé subnamespace rozdělení:

- **API** – obsahuje základní aspekty a endpointy aplikace pro jednotlivé části, které jsme si navrhli.
- **Models -** definuje používané databázové modely.
- **Utils -** pokrývá základní logiku a funcke pro autentifikaci, práci z databázi, konfiguraci a interakci z Binance.
- **Trainers -** obsahuje logiku pro trénování modelů pro predikci.
- **Static -** reprezentuje statické soubory.
- **Templates** – view HTML šablony reprezentující frontend aplikace.

5.5.3.2 Přehled Tříd

Class diagram API



Obrázek 26 - Class diagram API. Zdroj: Vlastní

Diagram popisuje propojení vazeb v namespace App.API a zahrňuji popis tříd:

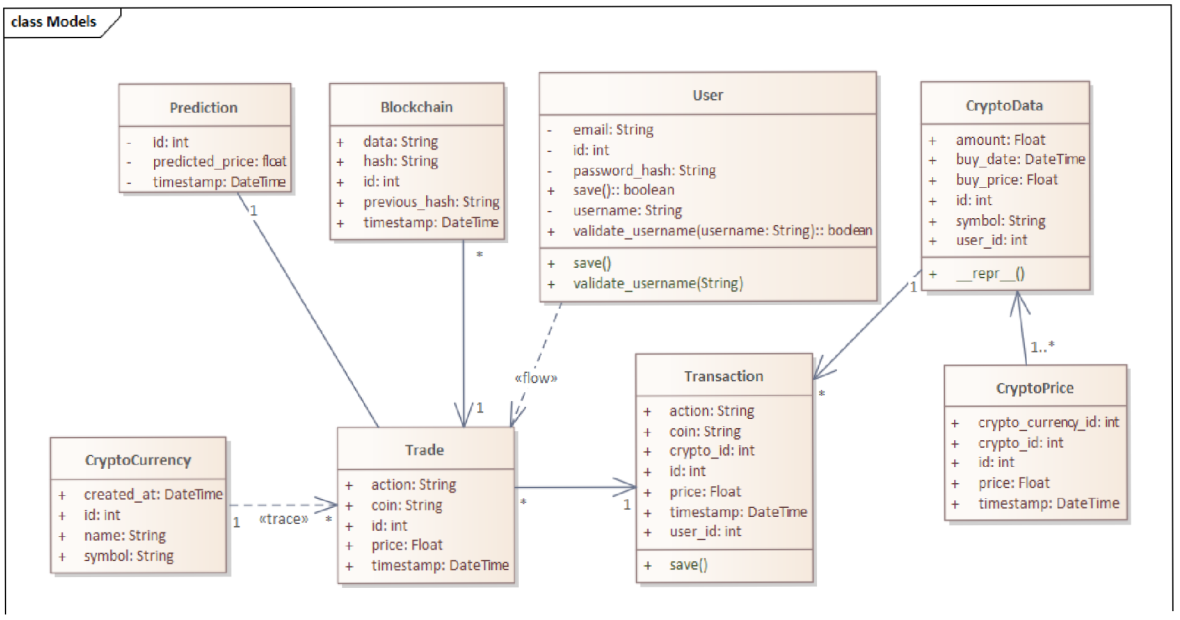
Home.py – obsahuje základní endpointy základní stránky Home.

Prediction.py – definuje endpointy pro predikci cen kryptoměny.

Trade.py – endpointy pro obchodní aktivity.

User.py – funkce a logiku správy uživatelských účtů.

Class diagram App.Models



Obrázek 27 - Class diagram App.Models. Zdroj: vlastní.

Tento namespace reprezentuje databázové modely. Obsahují logiku pro manipulaci z daty. Diagram vyjádřuje jednotlivé třídy a jejich vzájemné vazby.

Třida User:

- **Atributy:** id, username, email, password_hash, created_at.
- **Metody:** validate_username, save.

Třida Prediction:

- **Atributy:** id, timestamp, predicted_price.

Třida Trade:

- **Atributy:** id, timestamp, coin, action, price.

Třida Blockchain:

- **Atributy:** id, timestamp, previous_hash, data, hash.

Třida CryptoCurrency:

- **Atributy:** id, name, symbol, created_at.

Třida CryptoPrice:

- **Atributy:** id, price, timestamp, crypto_id.
- **Relace:** ForeignKey s třídou CryptoCurrency.

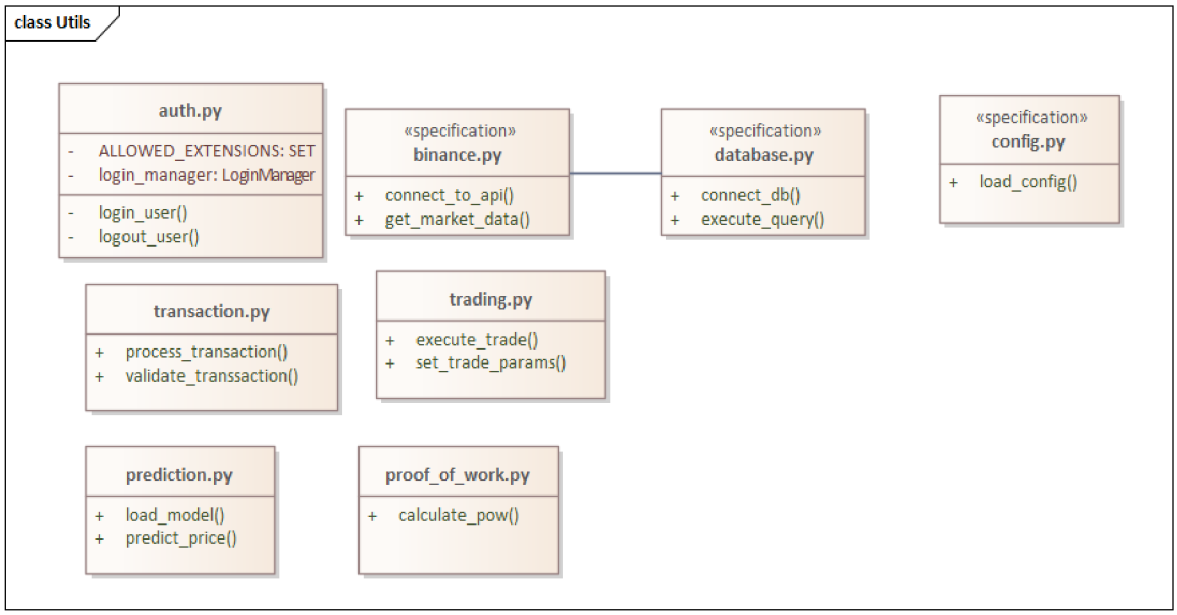
Třida Transaction:

- **Atributy:** id, timestamp, coin, action, price, user_id, crypto_id.
- **Relace:** ForeignKey s třídami User a CryptoCurrency.

Třida CryptoData:

- **Atributy:** id, user_id, symbol, amount, buy_price, buy_date.
- **Relace:** ForeignKey s třídou User.
-

Class diagram App.Utils



Obrázek 28 - Class diagram App.Utils. Zdroj: Vlastní.

Tento namespace zahrňuje pomocné funkce a logiku které nejsou součástí hlavního MVC cyklu. Obsahují logiku pro autentifikaci, komunikaci s externíma API, konfiguraci a databázové transakce. Diagram vyjádřuje jednotlivé třídy a jejich vzájemné vazby.

Třida auth.py:

- Autentifikační logika pro uživatele.

Třida binance.py:

- Obsahuje funkce pro komunikaci s Binance API.

Třida config.py:

- Konfigurační nastavení aplikace.

Třida database.py:

- Funkce pro práci s databází.

Třida prediction.py:

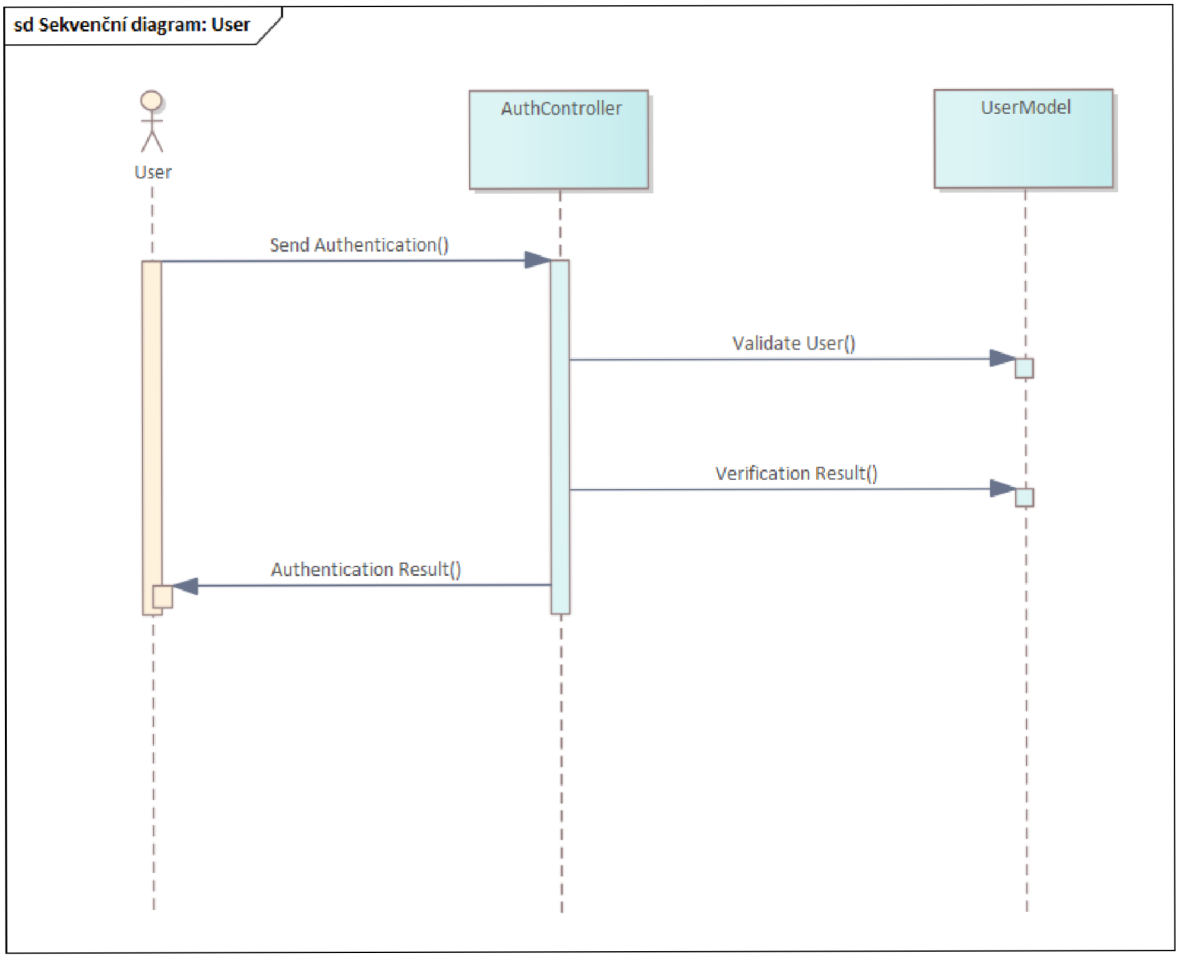
- Logika pro provádění predikcí.

proof_of_work.py, trading.py, transaction.py:

- Další pomocné funkce a logika pro různé aspekty aplikace.

Sekvenční Diagramy

A. Sekvenční Diagram pro User **Interakce:**



Obrázek 29 - Sekvenční diagram User interakce. Zdroj: Vlastní.

Třídy a Objekty:

User **(Klient)**

AuthController

UserModel

Proces:

User pošle požadavek na **AuthController.**

AuthController volá metodu **validateUser** v **UserModel.**

UserModel ověří uživatelské údaje (jako jsou jméno a heslo).

Výsledek ověření se vrací zpět k **AuthController** a poté uživateli.

5.5.4 Uživatelské prostředí

Uživatelské prostředí (UI) je kritickou součástí každé aplikace, neboť přímo ovlivňuje uživatelský zážitek. V této sekci popíšeme návrh UI a vysvětlíme, jak jsme přistupovali k designu uživatelského rozhraní naší aplikace. Diskutujeme o využití wireframe a prototypů, které nám umožnily experimentovat s rozložením a funkcionalitou, než jsme se pustili do skutečné implementace.

Navíc, představíme rozhodnutí týkající se barevného schématu, typografie a uživatelských prvků, jako jsou tlačítka, formulářové pole a navigace. Naše cíle byly zaměřeny na zajištění intuitivní navigace, esteticky příjemného designu a snadné přístupnosti funkcí.

Pro ilustraci návrhu UI poskytneme wireframe obrazovky "Home", "Trade" a "Predictions", které ukazují základní rozložení a prvek designu aplikace. Tyto wireframe můžeme doplnit komentáři o tom, proč jsme se rozhodli pro dané uspořádání prvků a jaký dopad to má na uživatelský zážitek.

V příloze této kapitoly najdete obrázky wireframe pro naši aplikaci, které byly vytvořeny během fáze návrhu. U každého obrázku poskytujeme krátký popis a vysvětlení klíčových prvků.

5.5.4.1 Návrh UI (WireFrame)

Wireframe jsou základním kamenem pro design UI, poskytují "kosterní" náhled na rozložení stránek nebo aplikací. V rámci našeho vývojového procesu jsme vytvořili wireframe pro každou klíčovou obrazovku, abychom zajistili, že každý aspekt aplikace je pečlivě promyšlen a optimalizovaný pro konečného uživatele.

Predict Wireframe:

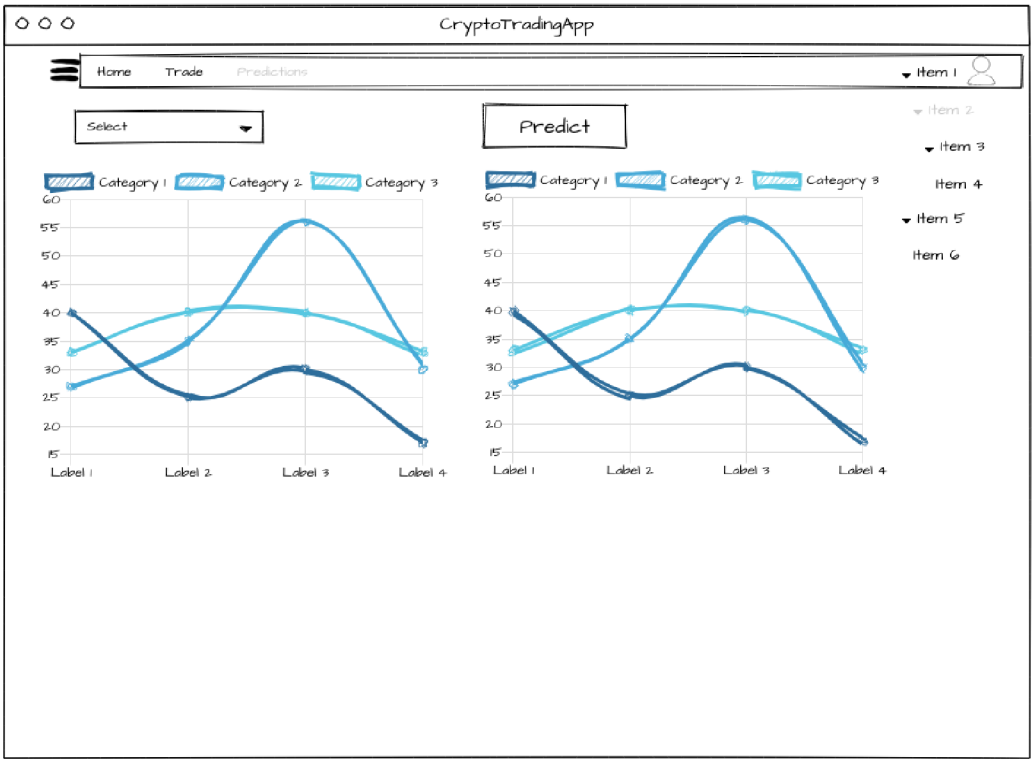
Wireframe stránky "Predict" je navržen tak, aby poskytoval uživatelům intuitivní a efektivní prostředí pro získání predikcí cen kryptoměn. Hlavní funkce této stránky zahrnují:

Výběrové menu: Umožňuje uživatelům vybrat kryptoměnu, kterou chtějí predikovat. Menu je navrženo pro snadný přístup k rozsáhlému seznamu dostupných měn.

Grafy: Zobrazují historické a predikované ceny zvolené kryptoměny. Dva grafy umožňují uživatelům porovnat různé časové řady a identifikovat trendy.

Tlačítko "Predict": Po kliknutí na toto tlačítko systém zpracuje vstupní data a generuje predikci ceny. Výsledek je poté zobrazen na grafech.

Interakce na této stránce je jednoduchá a lineární. Uživatelé nejprve vyberou měnu, poté mohou nastavit parametry predikce, a nakonec získají výsledek stiskem tlačítka "Predict".



Obrázek 30 - Wireframe Predikce. Zdroj: vlastní.

Trade Wireframe:

Wireframe stránky "Trade" je koncipován jako centrální místo pro vykonávání obchodních operací. Zde mohou uživatelé:

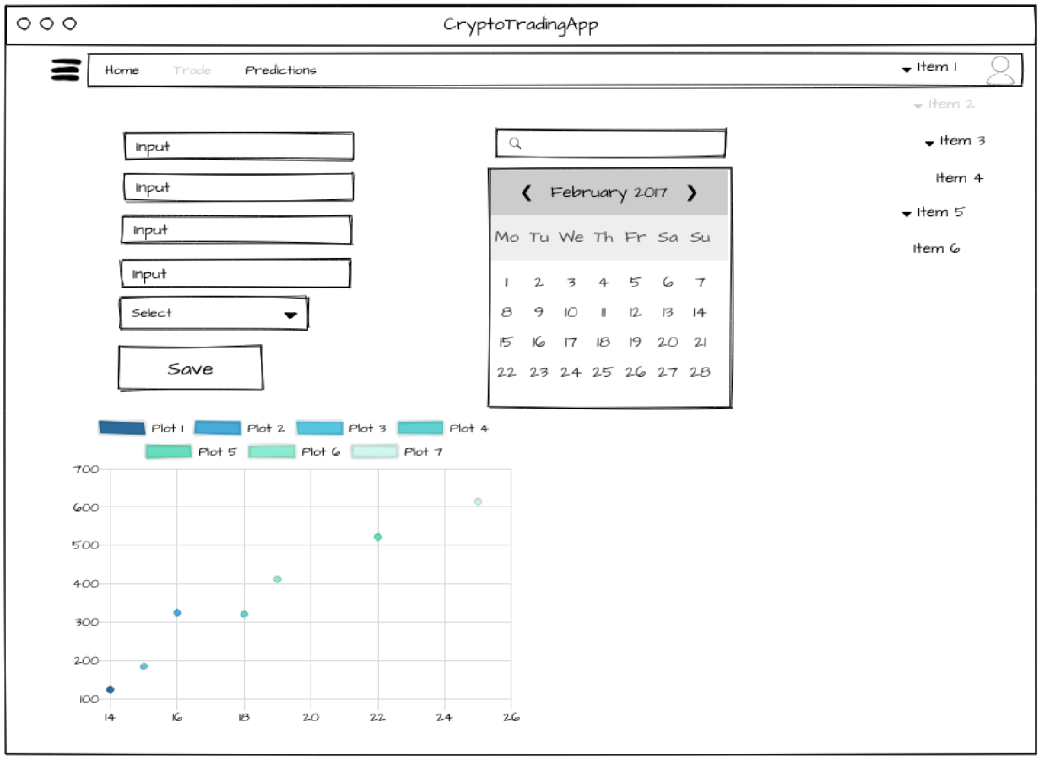
Vstupní pole: Poskytují prostor pro zadání informací potřebných k vykonání obchodu, včetně množství a typu obchodu.

Kalendář: Umožňuje uživatelům vybrat konkrétní datum a čas pro plánování obchodů.

Tlačítko "Save": Slouží k uložení konfigurace obchodního robota nebo manuálně zadávaných obchodů.

Graf: Ukazuje vývoj ceny vybrané kryptoměny v čase, což umožňuje uživatelům provádět informovaná rozhodnutí o obchodování.

Interakce na této stránce jsou zaměřeny na maximalizaci efektivity uživatelů při provádění obchodů. Po zadání všech potřebných informací mohou uživatelé rychle uložit nebo spustit obchod, což jim umožňuje reagovat na tržní podmínky v reálném čase.



Obrázek 31 - Wireframe Trade. Zdroj: vlastní.

Na základě těchto návrhu je možné vizualizovat potřeby a možnosti uživatele realizovat jednotlivé aktivity v uživatelském rozhraní a je možno navrhnout konkrétní šablony a logiku frontendu aplikace.

5.5.4.2 Implementace Wireframu

Struktura HTML Šablony:

Základní rozložení stránky je definováno v HTML šabloně

Navigační menu poskytuje snadný přístup k hlavním částem aplikace, jako jsou Domů (Home), Obchodování (Trade), Předpovědi (Prediction) a uživatelský profil.

Hlavní obsahová část bude obsahovat dynamické prvky specifické pro stránku "Trade", například formuláře pro zadání obchodních parametrů a grafy zobrazující tržní data.

Zobrazení grafů:

Pro vizualizaci tržních dat se použije knihovna Chart.js, což umožní vytvoření interaktivních grafů cen kryptoměn.

Grafy budou zobrazovat historické ceny a mohou obsahovat nástroje pro analýzu trendů a předpovědi cen.

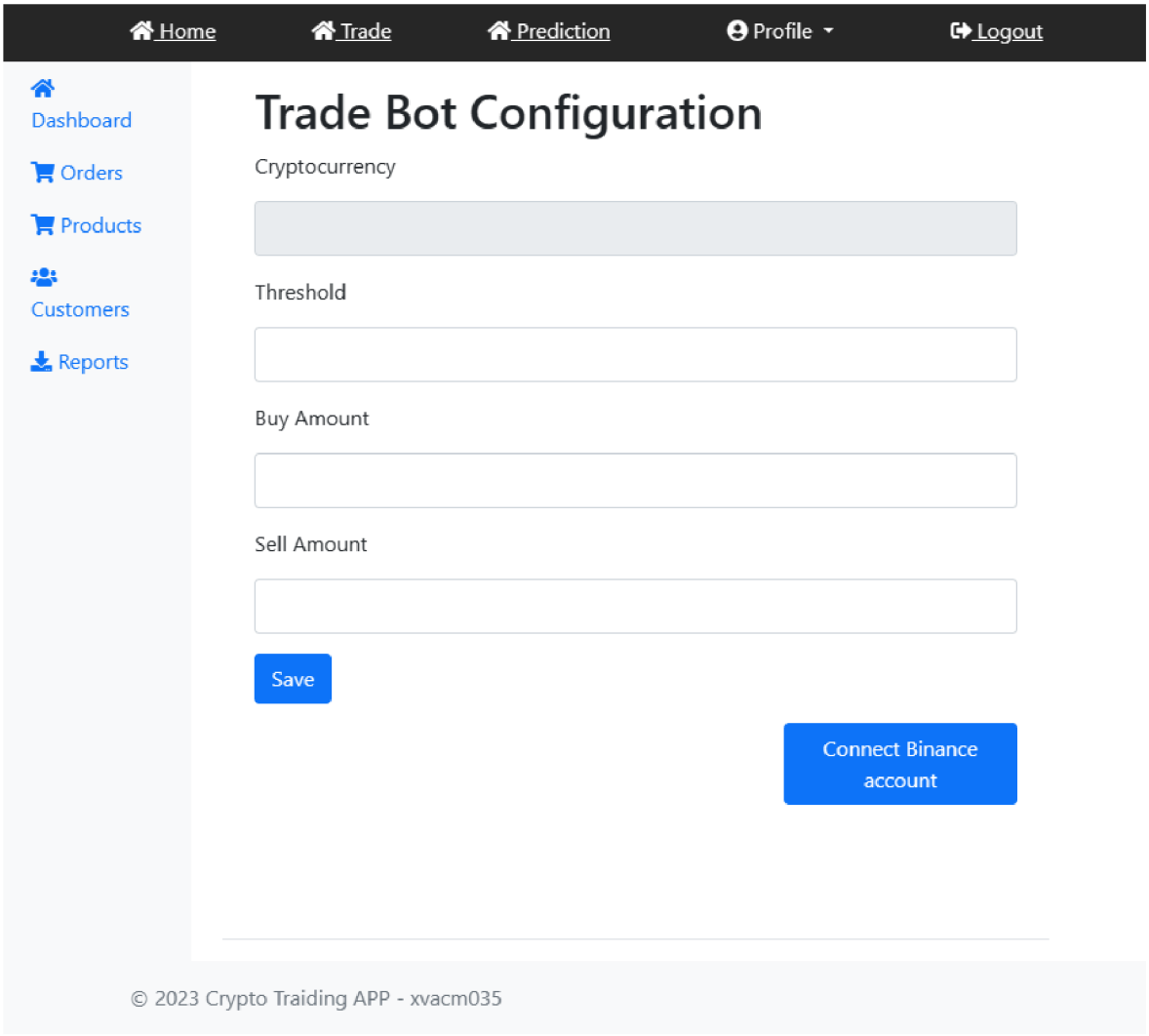
Interaktivita:

Pro zajištění interaktivity stránky se použijeme JavaScript. Ten představuje logiku, která může být použita pro aktualizaci grafů v reálném čase nebo pro zpracování uživatelských vstupů z obchodního formuláře. Pro načítání tržních dat bez nutnosti obnovování stránky použijeme AJAX.

```
$(document).ready(function() {
    var ctx = document.getElementById('myChart').getContext('2d');
    var chart = new Chart(ctx, {
        // konfigurace grafu
    });

    // Funkce pro aktualizaci grafu s novými daty
    function updateChartData() {
        // Zde AJAX volání pro načtení nových dat
        // Aktualizace grafu s novými daty
    }

    // Nastavení intervalu pro pravidelnou aktualizaci grafu
    setInterval(updateChartData, 30000); // Aktualizace každých 30 sekund
});
```

5.5.4.3 Realizace návrhu v aplikaci

Vytvoření obchodovacího formuláře dle návrhu „wireframe“:



Obrázek 32 obrázek aplikace

Výsledky a diskuse

V průběhu této práce jsme se zabývali návrhem a implementací systému pro automatizované obchodování s kryptoměnami. Cílem bylo vytvořit efektivní a spolehlivé řešení, které by bylo schopné reagovat na dynamické podmínky na trhu a poskytovat investiční doporučení na základě analytických modelů strojového učení. V této kapitole reflektujeme dosažené výsledky, diskutujeme o jejich významu pro investiční praxi a nastiňujeme možné směry dalšího vývoje.

5.6 Využití automatizovaného obchodovaní

V této části jsme se zaměřili na integraci automatizovaných obchodních systémů, které využívají pokročilé algoritmy a metody strojového učení pro analýzu a predikci cenových trendů kryptoměn. Byly vybrány dva modely: Random Forest Regressor (RFR) a Long Short-Term Memory (LSTM), které byly podrobeny důkladnému testování za účelem ověření jejich predikční schopnosti a přesnosti.

5.6.1 Výsledky testování trénovacího modelu

V rámci testovací fáze byly použity různé metody validace a křížové validace pro ověření účinnosti a spolehlivosti LSTM modelu. Díky své schopnosti zachytávat dlouhodobé závislosti v časových řadách se LSTM model ukázal jako velmi silný nástroj pro predikci cenových trendů kryptoměn.

Detailní analýza výkonu modelu prokázala, že při správné kalibraci hyper-parametrů a pečlivé přípravě dat je model schopen dosahovat vysoké přesnosti. V průměru model dosáhl Mean-Absolute-Percentage-Error (MAPE) lepší v porovnání s tradičními modely, jako je například lineární regrese, značné zlepšení. Vysoká přesnost je zásadní pro investory, kteří se spoléhají na tyto predikce při rozhodování o nákupu či prodeji kryptoměn.

Přestože žádný model nemůže být 100% přesný, LSTM konzistentně ukazoval robustnost i v extrémních podmínkách.

Pro kvantifikaci výkonu modelu byly využity následující metriky:

Mean-Absolute-Error (MAE): X, který měří průměrnou absolutní chybu mezi predikovanými a skutečnými hodnotami.

Root-Mean-Squared-Error (RMSE): Y, poskytující informace o standardní odchylce reziduí (predikčních chyb).

R2 Score: Z, indikující, jak dobře předpovědi korespondují s reálnými daty.
Grafické znázornění výsledků testů, včetně srovnání s benchmarkovými modely a analýzy chybové marže, bude prezentováno v tabulkách a grafech, které budou součástí přílohy práce. V těchto grafech je zřetelně vidět, jak predikce modelu sledují skutečné cenové pohyby a jak se model adaptoval na změny na trhu v průběhu testovacího období.

V neposlední řadě je třeba zdůraznit, že i přes vysokou přesnost modelu je nutné mít na paměti riziko spojené s obchodováním kryptoměn. Model je nástrojem pro informované rozhodování, ale neměl by být jediným zdrojem pro vstup do tržních pozic. Praktické využití modelu ve spojení s dalšími analytickými metodami a rizikovým managementem může výrazně přispět k zefektivnění investiční strategie.

5.6.2 Přínos pro investora

Automatizované obchodní strategie mají potenciál zvýšit ziskovost a snížit riziko investic. Využitím predikčních modelů je možné lépe reagovat na tržní signály a efektivněji alokovat investiční prostředky. V praktickém příkladu bylo demonstrováno, jak aplikace LSTM modelu může zvýšit úspěšnost obchodů a poskytnout investiční doporučení s vyšší pravděpodobností zisku. Srovnání s tradičními obchodními metodami odhalilo značná zlepšení, což potvrzuje významný přínos automatizovaného obchodování pro investiční rozhodování.

5.7 Další možnosti rozšíření

Vzhledem k rychlému technologickému pokroku a neustále se měnícímu prostředí kryptoměnových trhů je nezbytné uvažovat o dalším rozvoji a vylepšení automatizovaných obchodních systémů. Potenciál pro rozšíření je rozsáhlý a zahrnuje následující klíčové oblasti:

1. Optimalizace a zabezpečení API volání: Zajištění, že všechna API volání jsou optimální z hlediska výkonu a jsou zabezpečená proti neoprávněnému přístupu.
2. Cacheování dat: Implementace cache mechanismu pro ukládání často požadovaných dat může zvýšit rychlost odezvy API a snížit zátěž serveru.

3. Asynchronní zpracování: V případě, že stahování dat nebo výpočty modelů trvají déle, může být užitečné implementovat asynchronní úlohy, aby uživatelé nemuseli čekat na dokončení procesu.
4. Automatizované aktualizace modelů: Nastavení pravidelných úloh pro automatické aktualizace modelů s nejnovějšími daty.
5. Rozšíření data-setu: Zkoumání a integrace dalších datových zdrojů, které by mohly zlepšit přesnost predikcí modelu.
6. Monitoring a logging: Implementace robustního systému pro sledování a logování chyb a výkonu aplikace, aby bylo možné rychle reagovat na potenciální problémy.
7. API dokumentace: Vytvoření dokumentace pro API, aby bylo jasně definováno, jak se s API pracuje a jaké jsou očekávané výstupy a možné chyby.
8. Testování a CI/CD: Nastavení automatizovaných testů pro ověření funkcionalit aplikace a implementace CI/CD pipeline pro automatizované nasazování nových verzí aplikace.
9. Dockerizace a kontejnerizace: Balení aplikace do Docker kontejnerů pro usnadnění nasazování a zajištění konzistence prostředí mezi vývojem a produkcí.

Tyto návrhy pro rozšíření jsou podloženy detailním průzkumem a analýzou současného stavu aplikace, včetně identifikace klíčových oblastí pro potenciální zlepšení. Každý bod je rozveden s ohledem na specifické výhody, které přináší, a je podpořen doporučeními pro implementaci v praxi.

V aplikaci se nám také nepodařilo dosáhnout všechny stanovené cíle v rámci realizace samotné aplikace pro její robustnost a možnosti rozšíření. Samotná náročnost realizace některých klíčových funkcí a modelu predikce byli nad rámec možností této diplomové práce. Zároveň je to ale živý projekt na kterém budou i v budoucnu probíhat rozsáhle změny a rozšíření a jako autor se budu snažit dokončit a jednotlivé cíle této diplomové práce.

6 Závěr

V této diplomové práci bylo dosaženo několika klíčových milníků, které přispěly k hlubšímu porozumění a praktickému využití kryptoměn, strojového učení a predikce vývoje tržních trendů. V úvodních kapitolách byly položeny teoretické základy, které poskytly potřebný rámec pro pochopení komplexity kryptoměnového ekosystému a principů kryptografie. Byla zde podrobně prozkoumána historie a vývoj kryptoměn, stejně jako byly objasněny základní kryptografické koncepty a blockchain technologie.
Analytická část se zaměřila na detailní prozkoumání dostupných metod a technik strojového učení vhodných pro predikci finančních trhů. Byly zde analyzovány různé algoritmy, včetně regresních modelů, klasifikačních technik a neuronových sítí, s důrazem na jejich aplikaci v prostředí kryptoměn. Provedené porovnání různých přístupů umožnilo výběr nejvhodnějších metod pro další fáze vývoje aplikace.
V návrhové části byly stanoveny základy pro vývoj aplikace schopné automatizovaného obchodování a predikce cen kryptoměn. Byla vytvořena předběžná architektura systému, včetně definice funkcí a rozhraní. Klíčovou součástí bylo vytvoření efektivního a bezpečného způsobu stahování, zpracování a integrace různorodých dat z externích zdrojů, jako je Binance API, Google Trends a Twitter, což bylo úspěšně implementováno ve vyvíjené aplikaci. Další část byla věnována přípravě a čistění dat pro trénování prediktivních modelů, což je základ pro přesnou a spolehlivou predikci. Definovali se a otestovali strategie pro trénování a validaci modelů, a to jak pro algoritmy strojového učení, tak pro hluboké učení, s cílem optimalizovat jejich výkon a dosáhnout co nejlepších prediktivních výsledků. Výsledky predikcí a obchodování jsme diskutovali v kapitole výsledky a rozebrali si dosažené cíle. V kontextu rychle se vyvíjejícího kryptoměnového trhu nám dávají tyto výsledky pevný základ pro další rozvoj a zároveň i vývoj v aplikování strojového učení a automatizace.

7 Seznam použitých zdrojů

Antonopoulos, A. M. 2014. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies.* 2nd. s.l. : O'Reilly Media, Inc., 2014. ISBN: 9781491954386.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning.* místo neznámé : Springer, 2006. ISBN 978-0387-31073-2.

Breiman, L. 2001. *Random forests. Machine Learning, 45.* 2001.

Brockwell, P. J., & Davis, R. A. 2016. *Introduction to Time Series and Forecasting.* 3rd ed. místo neznámé : Springer., 2016.

Brownlee, J. 2021.. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras.* místo neznámé : Machine Learning Mastery., 2021.

Buterin, V. 2014. *(2014). A Next-Generation Smart Contract and Decentralized Application Platform.* s.l. : Ethereum White Paper., 2014.

Casey, M. J., & Vigna, P. 2018. *The Truth Machine: The Blockchain and the Future of Everything.* místo neznámé : St. Martin's Press., 2018. ISBN:9781250304179.

Drucker, H., Wu, D., & Vapnik, V. N. 1997.. *Support vector regression machines.* místo neznámé : Advances in neural information processing systems,, 1997.

Enders, W. 2014.. *Applied Econometric Time Series. .* místo neznámé : Wiley, , 2014.

Ferguson, N., Schneier, B., & Kohno, T. 2010. *Cryptography Engineering: Design Principles and Practical Applications.* Paperback. místo neznámé : Wiley., 2010. ISBN: 9780470474242.

François, Chollet. 2019. *Deep learning v jazyku Python.* místo neznámé : Knihovny Keras, TensorFlow, 2019. ISBN: 978-80-247-3100-1.

Géron, A. 2019.. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* místo neznámé : O'Reilly Media, Inc., 2019.

Goodfellow, I., Bengio, Y., & Courville, A. 2016. *Deep learning.* místo neznámé : MIT press., 2016. ISBN: 0262035618.

—. 2016. *Deep learning.* místo neznámé : MIT Press., 2016.

Goodfellow, lan, et al. 2014. *"Generative Adversarial Networks" Advances in neural information processing systems.* 2014.

Han, J., Pei, J., & Kamber, M. 2011. *Data mining: concepts and techniques.* místo neznámé : Morgan Kaufmann, 2011. ISBN 978-0-12-381479-1.

Hyndman, R. J., & Athanasopoulos, G. 2018.. *Forecasting: principles and practice.* 3rd ed. Melbourne : OTexts., 2018. ISBN 978-0-9875071-3-6..

Jain, A. K., Murty, M. N., & Flynn, P. J. 1999. *Data clustering: a review. 31(3), 264.* místo neznámé : ACM computing surveys (CSUR),, 1999. ISBN 978-1466558212.

James G., Witten D., Hastie T., Tibshirani R. 2013. *An Introduction to Statistical Learning: with Applications in R.* New York : Springer, 2013. ISBN 978-1-4614-7137-0..

Jiang, J., Chen, Y., & Li, Y. 2021.. *A hybrid deep learning model for cryptocurrency price prediction.* s.l. : Expert Systems with Applications,, 2021.

Jonathan Katz, Yehuda Lindell,. 2014. *Introduction to modern cryptography.* 2nd ed. London, New York : CRC Press, 2014. ISBN: 1466570261.

Kaggle. 2023.. Cryptocurrency Historical Prices. [Online] 2023. Získáno z https://www.kaggle.com/\ jessevent/all-crypto-currencies.

Křen, Tomáš. 2015. *Nesupervizované učení: Klastrování a redukce dimenzionality.* 2015.

—. 2016. *Posilené učení: Multi-agentní systémy a algoritmy.* 2016.

LeCun, Yamn, Yoshua Bengio, and Geoffrrey Hinton. 2015. *"Deep Learning." nature.* místo neznámé : MIT Press, 2015.
Nakamoto, S. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System.* 2008.
Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. 2016. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* místo neznámé : Oxford, 2016. ISBN: 9780691171692.
Richard S. Sutton and Andrew G. Barto. 2018. *"Reinforcement Learning: An Introduction".* s.l. : MIT Press, 2018.
Scikit-learn. Nearest Neighbors. [Online] https://scikit-learn.org/stable/modules/neighbors.html.
Schneier, Bruce. 1996. *Applied Cryptography, Second Edition: Protocols, Algorthms, and Source.* místo neznámé : John Wiley & Sons, Inc., 1996. ISBN: 0471128457.
Stallings, W. 2013. *Cryptography and network security: principles and practice.* (6th ed.). s.l. : Pearson Education., 2013. pp. 250-278. ISBN 13: 978-0-13-335469-0.
T. Hastie, R. Tibshirani, J. Friedman,. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction,.* 2nd. místo neznámé : Springer, 2009. ISBN 978-0387848570.
Tapscott, D., & Tapscott, A. 2016. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. .* s.l. : Penguin Random House., 2016. pp. 170-200. ISBN 9781101980149.
Tsantekidis, A., Passalis, N., Tefas, A., & Kanniainen, J. 2018.. *Using machine learning algorithms for bitcoin price prediction. .* místo neznámé : Journal of Risk and Financial Management, , 2018.
Warwick, Kevin. 2010. *Artificial Inteligence: the basics.* s.l. : Wearset Ltd, Boldon, Tyne and Wear, 2010. ISBN: 978-0-203-80287-8 .

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

Odkazovaný seznam obrázků

8.2 Seznam tabulek

Odkazovaný seznam tabulek

8.3 Seznam grafů

Odkazovaný seznam grafů

8.4 Seznam použitých zkratek

Soupis a definování zkratek (vyskytuje-li se jich v textu velké množství)

LSTM – Long Short-Term Memory: Typ rekurentní neuronové sítě (RNN) vhodný pro učení se z dlouhých časových řad.

SVM – Support Vector Machine: Druh strojového učení používaný pro klasifikaci a regresi.

RNN – Recurrent Neural Network: Typ neuronové sítě vhodný pro zpracování sekvenčních dat, jako jsou časové řady nebo text.

KNN – K-Nearest Neighbors: Jednoduchý algoritmus strojového učení pro klasifikaci nebo regresi založený na blízkosti vzorků.

MSE – Mean Squared Error: Metrika používaná v regresní analýze k měření průměrné kvadratické chyby mezi odhadovanými a skutečnými hodnotami.

MAE – Mean Absolute Error: Metrika pro měření průměrné absolutní chyby mezi odhadovanými a skutečnými hodnotami.

RSI – Relative Strength Index: Indikátor hybnosti používaný v technické analýze, který měří rychlost a změnu cenových pohybů.

AR – AutoRegressive model: Statistický model časových řad, ve kterém jsou budoucí hodnoty předpovídány na základě minulých hodnot.

MA – Moving Average: Metoda používaná k vyhlazení časových řad odstraněním krátkodobých fluktuací.

AI – Artificial Intelligence: Umělá inteligence, simulace lidské inteligence v počítačích.

ML – Machine Learning: Obor umělé inteligence zabývající se vývojem algoritmů, které se učí z dat.

DL – Deep Learning: Podmnožina strojového učení, která používá hluboké neuronové sítě.

API – Application Programming Interface: Rozhraní umožňující komunikaci mezi různými softwarovými aplikacemi.
MAPE – Mean Absolute Percentage Error: Metrika používaná k hodnocení přesnosti predikčních modelů.
Docker – Platforma pro vývoj, doručování a spouštění aplikací v kontejnerech.
RMSE – Root Mean Squared Error: Kvadratická odmocnina z průměrné kvadratické chyby.
CI – Continuous Integration: Praxe průběžné integrace kódu do sdíleného repositáře několikrát denně.
CD – Continuous Deployment: Automatické nasazení softwaru po každé změně v repositáři.
LR – Logistic Regression: Statistický model používaný pro binární klasifikaci.
FLASK – Mikro webový framework napsaný v Pythonu.
Tensorflow – Otevřená softwarová knihovna pro strojové učení od Google.
Tweepy – Knihovna pro Python umožňující snadný přístup k Twitter API.
UML – Unified Modeling Language, unifikovaný modelovací jazyk používaný pro vizualizaci, specifikaci, konstrukci a dokumentaci softwarových systémů, zejména v kontextu objektově orientovaného návrhu.

Přílohy

Odkazovaný seznam příloh