EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2025/36124048429843204

ZBER ÚDAJOV A PREDIKCIA VYBRANÝCH ČASOVÝCH RADOV VALUTOVÝCH KURZOV

Diplomová práca

2025 **Bc. Elmira Voloshchuk**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

ZBER ÚDAJOV A PREDIKCIA VYBRANÝCH ČASOVÝCH RADOV VALUTOVÝCH KURZOV

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Ekonómia a manažment

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: doc. Ing. Jaroslav Kultan, PhD., PhD.

2025 **Bc. Elmira Voloshchuk**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Elmira Voloshchuk
Študijný program: informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Inžinierska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Zber údajov a predikcia vybraných časových radov valutových kurzov.

Anotácia: Jedným z predpokladov úspešnej činnosti podniku je odhad zmeny ceny surovín a výrobkov v dôsledku zmeny výmenného kurzu jednotlivých mien. Preto je vhodné poznať jednotlivé súčasné hodnoty, predchádzajúce hodnoty a odhadnúť s najväčšou pravdepodobnosťou kurz na budúce obdobie. Preto, pre účely podniku, je potrebné vytvoriť systém, ktorý bude, na základe získaných údajov z centrálnych národných bánk viacerých štátov, odhadovať budúcu hodnotu vybraného kurzu a zobrazovať ho na stránke organizácie. Vytvorený informačný systém je zameraný na priebežný zber údajov z vybraných internetových stránok, uloženie do databázy, výpočet alebo adaptáciu modelu a zobrazenie predikcie údajov na ďalšie obdobie.

Vedúci: doc. Ing. Jaroslav Kultan, PhD.
Katedra: KAI FHI - Katedra aplikovanej informatiky
Vedúci katedry: doc. Ing. Mgr. Peter Schmidt, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 09.03.2024

Dátum schválenia: 11.03.2024

prof. Ing. Ivan Brezina, CSc.
osoba zodpovedná za realizáciu študijného programu

ABSTRAKT

VOLOSHCHUK, Elmira: *Zber údajov a predikcia vybraných časových radov valutových kurzov*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky. Katedra aplikovanej informatiky – Vedúci záverečnej práce: doc. Ing. Jaroslav Kultan, PhD., PhD. – Bratislava: FHI EUBA, 2025, 105 s.

Hlavným cieľom tejto diplomovej práce je návrh a implementácia informačného systému, ktorý umožňuje automatický zber údajov o valutových kurzoch, ich ukladanie do databázy a predikciu budúcich hodnôt s využitím štatistických a moderných modelov strojového učenia. Systém je navrhnutý na integráciu skutočných údajov z Európskej centrálnej banky, ich spracovanie a vizualizáciu predikčných výsledkov prostredníctvom webovej aplikácie, čím slúži ako praktický nástroj na podporu ekonomického rozhodovania v reálnom čase.

Záverečná práca pozostáva z 105 strán, vrátane 11 tabuliek a 61 obrázkov, a je rozdelená do štyroch kapitol. Prvá kapitola sa zaoberá teoretickým preskúmaním problematiky zberu údajov, analýzou časových radov a modelmi predikcie. Druhá kapitola definuje hlavný cieľ a čiastkové ciele práce. Tretia kapitola predstavuje použitú metodológiu, vrátane výberu technológií na zber údajov a modelovania časových radov, ako aj implementácie systému s použitím PostgreSQL a Django. Štvrtá kapitola sa zameriava na prezentáciu výsledkov, kde sú porovnané rôzne predikčné modely na základe presnosti a spoľahlivosti, hodnotené pomocou štandardných metrík, ako sú MAE a MSE.

Kľúčové slová: kurzy valút, matematický model, predikcia časových radov, strojové učenie, vizualizácia dát, internetová aplikácia, PostgreSQL, Django

ABSTRACT

VOLOSHCHUK, Elmira: *Data collection and prediction of selected time series of currency rates*. – University of Economics in Bratislava. Faculty of Economic Informatics. Department of Applied Informatics – Thesis Supervisor: doc. Ing. Jaroslav Kultan, PhD., PhD. – Bratislava: FHI EUBA, 2025, 105 pages.

The primary objective of this thesis is to design and implement an information system that enables the automated collection of actual currency exchange rate data, their storage in a database, and the prediction of future values using statistical and modern machine learning models. The system is designed to integrate real data from the European Central Bank, process it, and visualize prediction results through a web application, thereby serving as a practical tool to support economic decision-making in real time.

The thesis consists of 105 pages, including 11 tables and 61 images, and is divided into four chapters. The first chapter deals with the theoretical exploration of data collection, time series analysis, and prediction models. The second chapter defines the main and specific objectives of the thesis. The third chapter presents the applied methodology, including the selection of technologies for data collection, time series modeling, and the system implementation using PostgreSQL and Django. The fourth chapter focuses on presenting the results, comparing various predictive models based on accuracy and reliability, evaluated using standard metrics such as MAE and MSE.

Keywords: currency exchange rates, mathematical model, time series prediction, machine learning, data visualization, web application, PostgreSQL, Django

Poďakovanie

Touto cestou by som sa chcela poďakovať vedúcemu diplomovej práce doc. Ing. Jaroslavovi Kultanovi, PhD., PhD. za odborné konzultácie, cenné rady a usmerňovanie pri vypracovaní diplomovej práce.

OBSAH

Zoznam obrázkov a tabuliek

Zoznam tabuliek

Zoznam vzorcov

Zoznam skratiek

API	Application Programming Interface (Aplikačné programové rozhranie)
ARIMA	AutoRegressive Integrated Moving Average (AutoRegresný integrovaný kĺzavý priemer)
CNY	Čínsky jüan
CZK	Česká koruna
EDA	Exploratory Data Analysis (Exploratívna analýza dát)
ECB	European Central Bank (Európska centrálna banka)
ES	Exponential Smoothing (Exponenciálne vyrovnávanie)
GBP	Britská libra
GPT	Generative Pretrained Transformer (Generatívny predtrénovaný transformer)
HUF	Maďarský forint
IoT	Interner og Things (Internet vecí)
JSON	JavaScript Object Notation (JavaScriptová notácia objektov)
MAE	Mean Absolute Error (Priemerná absolútna chyba)
MC	Markov Chain (Markovov reťazec)
MSE	Mean Squared Error (Priemerná kvadratická chyba)
ORM	Object-Relational Mapping (Objektovo-relačné mapovanie)
PLN	Poľský zlotý
RNN	Recurrent Neural Network (Rekurentná neurónová sieť)
RPA	Robotic Process Automation (Automatizácia robotických procesov)
RMSE	Root Mean Squared Error (Kvadratická stredná chyba)
LSTM	Long Short-Term Memory (Sieť s dlhodobou a krátkodobou pamäťou)
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
SDMX	Statistical Data and Metadata eXchange
SSIS	SQL Server Integration Services (Služby integrácie so SQL Serverom)
USD	Americký dolár
VAR	Vector Autoregressive Model (Vektorový autoregresný model)
XML	eXtensible Markup Language

Úvod

Predikcia zohráva kľúčovú úlohu v každodennej ľudskej činnosti a pri rozhodovaní o budúcnosti – napríklad predpovedanie počasia a rôznych prírodných javov, plánovanie výrobných činností a predaja tovaru, prognózy súvisiace so správaním finančného trhu možno zaradiť medzi tie príklady, pri ktorých presnejšie hodnotenia určujú špecifiká prijatých opatrení a majú významný vplyv na prípravu plánov konkrétnych scenárov správania v budúcnosti.

Jednou z dôležitých úloh v oblasti predpovedí je predikcia výmenných kurzov. Devízový trh je dôležitou súčasťou globálnej ekonomiky a menové výkyvy majú významný vplyv na podnikanie, investície a každodenné finančné rozhodnutia. Predikcia výmenných kurzov napomáha minimalizovať riziká a prijímať informovanejšie rozhodnutia pre súkromných investorov aj profesionálnych hráčov na finančnom trhu.

Táto práca sa zaoberá problémom predikcie časových radov výmenných kurzov voči euru. Relevantnosť tejto práce je obzvlášť veľká v krajinách, kde je euro oficiálnou menou, ako je Slovensko.

Okrem ekonomických a obchodných cieľov možno predikciu výmenných kurzov použiť aj na analýzu makroekonomických trendov a vplyvu rôznych faktorov, ako je politická nestabilita, inflácia a úrokové sadzby na hodnotu mien. Presné predpovede výmenných kurzov môžu pomôcť vláde a finančným inštitúciám plánovať a rozvíjať hospodársku politiku.

Cieľom tejto práce je nielen teoreticky preskúmať dostupné prístupy k zberu a predikcii údajov, ale aj implementovať konkrétny informačný systém, ktorý bude schopný zhromažďovať údaje o valutových kurzoch v reálnom čase, ukladať ich do databázy a využívať ich na predikciu budúcich hodnôt. Vytvorený systém bude založený na kombinácii štatistických metód a moderných techník strojového učenia, pričom výsledky predikcie budú vizualizované vo forme prehľadných grafov na webovej platforme.

Je dôležité poznamenať, že predpovedanie meny je zložitý proces, keďže výmenné kurzy sú ovplyvňované mnohými faktormi vrátane ekonomických ukazovateľov, geopolitických udalostí, menovej politiky a nálady na trhu. Presnosť predpovedí sa môže líšiť a vždy podlieha neistote a neočakávaným udalostiam. Aj napriek svojej komplexnosti zostáva predikcia výmenných kurzov neoddeliteľnou súčasťou nástrojov, ktoré pomáhajú znižovať neistotu a lepšie sa pripraviť na budúci vývoj.

1 Súčasný stav riešenej problematiky doma a v zahraničí

Zber, analýza a predikcia údajov o výmenných kurzoch predstavujú kľúčové úlohy v oblasti finančného riadenia a rozhodovania. Systémové spracovanie týchto dát si vyžaduje nástroje na zber údajov v reálnom čase, ako aj ich následnú analýzu a modelovanie. Medzi hlavné výzvy patrí zabezpečenie presnosti, spoľahlivosti a aktuálnosti získaných dát, ako aj výber vhodných matematických modelov a algoritmov na predikciu.

Teoretický a metodický základ, ktorý je predmetom prvej kapitoly, je východiskom pre návrh a implementáciu praktickej časti systému. Táto kapitola poskytuje prehľad o súčasných technológiách zberu dát, analýze časových radov a matematických metódach predikcie, ktoré sú základom pre tvorbu efektívnych riešení v oblasti spracovania údajov o výmenných kurzoch.

1.1 Technológie zberu a spracovania údajov v reálnom čase

V súčasnosti zaznamenávame celosvetový nárast využívania technológií na zber a spracovanie údajov v reálnom čase. Podľa správy IDC Worldwide Big Data and Analytics Spending Guide sa v roku 2023 očakáva, že globálne výdavky na riešenia Big Data a analýzu údajov dosiahnu viac ako 330 miliárd dolárov, čo predstavuje medziročný nárast o 15 %. V Európe sa tieto technológie aktívne zavádzajú vo finančnom sektore, kde 68 % spoločností využíva automatizované procesy na spracovanie údajov v reálnom čase (IDC, 2024).

Na Slovensku sú technológie na zber dát v reálnom čase čoraz populárnejšie, najmä v oblastiach, ako sú bankovníctvo, verejná správa a priemyselná výroba. Podľa správy Európskej komisie o digitalizácii Európy, približne 45 % podnikov na Slovensku využíva digitálne technológie vo svojich procesoch (European Commission, 2023). Slovenská sporiteľňa a ďalšie banky aktívne využívajú API, ako napríklad PSD2 API, na integráciu údajov z rôznych zdrojov, čím zlepšujú služby zákazníkom a optimalizujú svoje obchodné procesy.

V oblasti finančného sektora má zber údajov v reálnom čase zásadný význam pre monitorovanie a analýzu trhových trendov, hodnotenie rizík a optimalizáciu investičných stratégií. Schopnosť spracovávať veľké objemy údajov s minimálnym oneskorením dáva inštitúciám možnosť inštitúciám nielen efektívnejšie reagovať na aktuálne trhové podmienky, ale aj predikovať budúce výkyvy a adaptovať svoje stratégie v reálnom čase.

V reálnom svete sa často stretávame s výzvami, ako sú nekompletné údaje, legislatívne obmedzenia týkajúce sa ochrany osobných údajov a problémy s vzájomnou kompatibilitou a pretekaním dát (interoperabilitou) medzi rôznymi platformami. Tieto aspekty majú zásadný vplyv na úspešnosť implementácie zberu a spracovania údajov, pretože ich zvládnutie si vyžaduje kombináciu pokročilých technických a štatistických prístupov.

1.1.1 Web scraping

Web scraping je technika, ktorá umožňuje automatickú extrakciu informácií z webových stránok pomocou skriptov alebo špeciálneho softvéru, identifikuje požadované informácie a extrahuje ich do štruktúrovaného formátu, napríklad do tabuľky alebo databázy. Typické nástroje, ktoré sa používajú na web scraping, zahŕňajú Python knižnice ako BeautifulSoup, Scrapy alebo Selenium (Mitchell, 2018).

Existujú dva spôsoby extrakcie údajov z webových stránok, technika manuálnej extrakcie a technika automatizovanej extrakcie.

1. Techniky manuálnej extrakcie: Manuálne kopírovanie a prilepenie obsahu stránky patrí do tejto techniky. Aj keď je to zdĺhavé, časovo náročné a opakované, je to efektívny spôsob, ako odstrániť údaje zo stránok, ktoré majú dobré opatrenia proti zoškrabaniu, ako je detekcia botov.

2. Techniky automatizovanej extrakcie: Softvér na zoškrabovanie webu sa používa na automatickú extrakciu údajov zo stránok na základe požiadaviek používateľa.

- Analýza HTML: Analýza znamená urobiť niečo zrozumiteľným, aby sa to analyzovalo časť po časti. V podstate to znamená previesť informácie v jednej forme do inej formy, ktorá je jednoduchá, s ktorou sa ľahšie pracuje. Analýza HTML znamená prijatie kódu a extrahovanie relevantných informácií z neho na základe požiadavky používateľa. Vykonáva sa hlavne pomocou JavaScriptu, cieľom, ako už názov napovedá, sú stránky HTML.
- DOM Parsing: The Document Object Model je oficiálnym odporúčaním World Wide Web Consortium. Definuje rozhranie, ktoré umožňuje používateľovi upravovať a aktualizovať štýl, štruktúru a obsah dokumentu XML.
- Softvér na scraping webu: V súčasnosti je k dispozícii veľa nástrojov na scraping webu alebo sú prispôsobené na to, aby používatelia získali požadované informácie z miliónov webových stránok (GeeksforGeeks, 2023).

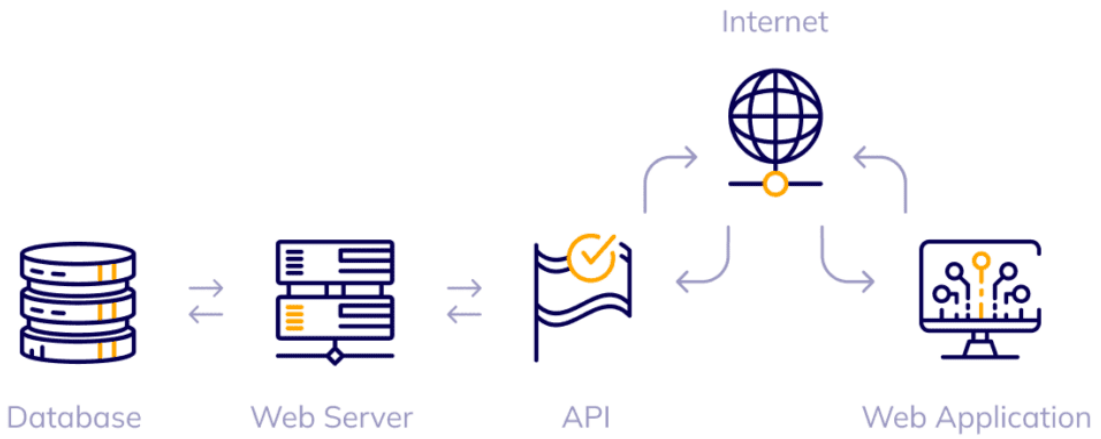
Vzhľadom na rozmach web scrapingu a jeho rôznorodé využitie sa čoraz častejšie otvára otázka jeho **legalizácie.**

1.1.2 Integrácia s API

Aplikačné programové rozhranie (Application Programming Interface, API) je súbor protokolov, definícií a nástrojov, ktorý umožňuje komunikáciu medzi dvoma softvérovými aplikáciami. API sú navrhnuté tak, aby poskytovali vrstvu abstrakcie, ktorá skrýva zložitosť implementácie a dátovej štruktúry, pričom sprístupňuje základnú funkcionalitu jednoduchým a prístupným spôsobom. API revolučným spôsobom zmenili zber údajov tým, že umožňujú používateľom prístup k službám alebo dátovým súborom bez potreby priameho prístupu k databázam alebo vnútorným mechanizmom daného systému.

API — je brána, ktorá umožňuje jednej aplikácii komunikovať s inými aplikáciami a definuje, ako táto komunikácia prebieha. Podporujú prepojenia medzi technológiami na zlepšenie používateľského zážitku. API je súbor softvérových funkcií a procedúr (Medjaoui et al., 2018).

API funguje ako rozhranie, ktoré umožňuje komunikáciu medzi rôznymi softvérovými komponentami, ako je znázornené na Obr. 1.



Obrázok 1: Fungovanie API
Zdroj: Qureshi, 2023

Proces začína v databáze, kde sú uchované všetky potrebné údaje. Webový server slúži ako medzičlánok, ktorý prijíma požiadavky od API a komunikuje s databázou, aby poskytol alebo aktualizoval informácie. API sprístupňuje tieto údaje aplikáciám tým, že spracúva požiadavky z webových aplikácií a sprostredkováva ich komunikáciu s webovým serverom. API komunikuje cez internet, čím umožňuje zdieľanie údajov medzi rôznymi systémami nezávisle od ich fyzickej polohy. Webová aplikácia alebo iný klient odosiela požiadavky cez

API, ktoré následne zabezpečí zisk údajov z databázy a vráti ich aplikácii na zobrazenie používateľovi. Takáto architektúra umožňuje flexibilnú a efektívnu výmenu informácií, čím podporuje automatizáciu a dynamickú integráciu údajov (Qureshi, 2023).

Existuje niekoľko typov API, ale na účely zberu údajov sa najčastejšie používajú REST (Representational State Transfer) a SOAP (Simple Object Access Protocol).

RESTful API využíva HTTP metódy ako GET, POST, PUT a DELETE na vykonávanie operácií s dátovými zdrojmi. V typickom scenári zberu údajov sa na API endpoint odošle požiadavka GET, ktorá následne poskytne požadovaný dataset v štruktúrovanom formáte, zvyčajne JSON (JavaScript Object Notation) alebo XML (eXtensible Markup Language). Použitie RESTful API v zbere údajov ponúka niekoľko výhod, vrátane škálovateľnosti, bezstavovosti a jednoduchosti integrácie s webovými aplikáciami (Fielding, 2000).

API sa využíva napríklad na integráciu platieb, prihlásenie do sociálnych sietí alebo prístup k verejným dátam (Flatlogic, 2024).

Pri implementácii integrácie s API sú bežne používané technológie a nástroje ako:

ETL a ELT nástroje:

- **ETL (Extract, Transform, Load)**: Používa sa na extrakciu dát z rôznych zdrojov, ich transformáciu a načítanie do cieľového systému.
- **ELT (Extract, Load, Transform)**: Moderná alternatíva, ktorá umožňuje rýchlejšiu prácu s veľkými dátovými objemami (Sivabalan, 2021).

SSIS (SQL Server Integration Services): SSIS umožňuje spracovanie a načítanie dát prostredníctvom robustných ETL nástrojov. Tento prístup je užitočný pre integráciu komplexných dátových zdrojov (Zulkifli Arsyad, 2021).

1.1.3 Zber dát pomocou nástrojov Microsoft Excel

Microsoft Excel je komplexný nástroj, ktorý okrem analýzy a vizualizácie údajov poskytuje aj možnosti automatizovaného získavania a spracovania dát z rôznych zdrojov.

1. VBA (Visual Basic for Applications) je skriptovací jazyk zabudovaný v aplikáciách balíka Microsoft Office, prostredníctvom ktorého možno automatizovať opakované úlohy, pripájať sa k externým zdrojom a spracúvať údaje.

VBA dokáže extrahovať údaje z webových stránok cez web scraping alebo prostredníctvom HTTP požiadaviek. Pomocou VBA môžete odosielať GET/POST požiadavky a získané údaje spracovať ako štruktúrované dáta. Skripty môžu automaticky prechádzať na URL adresy, sťahovať obsah stránky a extrahovať potrebné informácie.

VBA tiež umožňuje pripojenie k API rozhraniam a zber údajov v reálnom čase. Pomocou knižnice MSXML2.XMLHTTP je možné odosielať HTTP požiadavky na API a spracovávať odpovede vo formáte JSON alebo XML. API kľúče, autentifikácia a ďalšie parametre sa dajú nastaviť priamo v skripte.

Jednou z hlavných výhod VBA je jeho schopnosť prispôsobiť a optimalizovať Excel pre konkrétne potreby jednotlivých užívateľov či organizácií. Vďaka tomuto prispôsobeniu môžu používatelia automatizovať úlohy od základných výpočtov až po komplexné operácie, ako je tvorba vlastných funkcií, výpočet údajov v reálnom čase, automatické formátovanie alebo pokročilá analýza dát.

Bullen, Green a Bovey (2007) poukazujú na to, že práve prispôsobiteľnosť a možnosti automatizácie robia z VBA obľúbený nástroj medzi finančnými analytikmi, vedcami a ďalšími odborníkmi, ktorí potrebujú flexibilitu pri práci s dátami. Využitie VBA je najefektívnejšie práve pri opakujúcich sa úlohách, ktoré sa vykonávajú denne či týždenne, ako sú import údajov, generovanie pravidelných reportov alebo čistenie dát.

Napriek mnohým výhodám má VBA aj svoje obmedzenia. Pri práci s veľkými datasetmi môže byť VBA pomalšie a menej výkonné. Ďalšou nevýhodou je, že VBA si vyžaduje znalosti programovania, čo môže byť výzvou pre používateľov, ktorí nemajú skúsenosti s kódom alebo algoritmickým myslením. Na to, aby využili plný potenciál VBA, musia rozumieť základným programovacím konceptom, ako sú cykly, podmienky alebo deklarácia premenných, čo môže byť pre začiatočníkov náročné.

VBA sa využíva najmä v podnikových riešeniach a v situáciách, kde je potrebná automatizácia v rámci Microsoft Office (hlavne Excelu). Avšak v modernej dátovej analytike sa čoraz viac nahrádza Pythonom a inými robustnejšími riešeniami.

2. Power Query je nástroj na získavanie a transformáciu údajov, zabudovaný priamo v Exceli, ktorý umožňuje jednoduchý prístup k viacerým zdrojom dát vrátane webových stránok, databáz, súborov a API (Microsoft Documentation, 2023).

Jeho hlavnou výhodou je možnosť vykonávať pokročilú transformáciu dát cez grafické rozhranie bez nutnosti programovania, pričom kód v jazyku M sa generuje automaticky na pozadí. Power Query umožňuje získavať dáta priamo z HTML tabuliek a iných štruktúrovaných prvkov na webe. Priamo v prostredí Power Query môžete zadať URL adresu webovej stránky a prehliadať jednotlivé tabuľky alebo prvky, ktoré chcete importovať.

Tento nástroj dokáže pracovať aj s API, kde je prístup k údajom vo formáte JSON alebo XML. Na rozdiel od VBA je nastavenie jednoduchšie a Power Query dokáže automaticky rozbaliť JSON do tabuliek.

Jednou z jeho najväčších výhod je intuitívne rozhranie, ktoré umožňuje používateľom vykonávať komplexné transformácie a čistenie údajov bez potreby programovania. Vďaka vizuálnym nástrojom a jednoduchému používateľskému rozhraniu sa Power Query stáva prístupným pre širokú škálu používateľov, vrátane tých, ktorí nemajú pokročilé technické zručnosti. Ďalšou výraznou výhodou je možnosť automatizácie opakujúcich sa úloh; procesy ako import a čistenie dát môžu byť nastavené na automatické vykonávanie pri každej aktualizácii zdrojových dát, čím sa šetrí čas a eliminuje riziko chýb. Power Query je tiež flexibilný, čo sa týka zdrojov dát – podporuje pripojenie k databázam, webovým službám, cloudovým úložiskám a dokonca k ďalším aplikáciám Microsoftu. Navyše, Power Query je optimalizovaný pre prácu s veľkými datasetmi a v kombinácii s Power Pivot umožňuje ďalšie spracovanie veľkých objemov dát v Exceli.

Na druhej strane, Power Query má aj svoje nevýhody, jednou z nich je obmedzená podpora pre komplexné analytické funkcie, ktoré sú dostupné v špecializovaných programoch na analýzu dát, ako sú Python alebo R. Flexibilita Power Query pre personalizované transformácie je tiež obmedzená; náročnejšie transformácie si vyžadujú použitie jazyka M, ktorý je menej intuitívny a môže byť pre začiatočníkov náročný. Nakoniec, Power Query nie je vhodný na spracovanie dát v reálnom čase. Aktualizácia dát je obmedzená na manuálne alebo plánované intervaly, čo môže byť nevýhodou pre aplikácie, ktoré vyžadujú dynamické, živé dáta (Microsoft Documentation,, 2023).

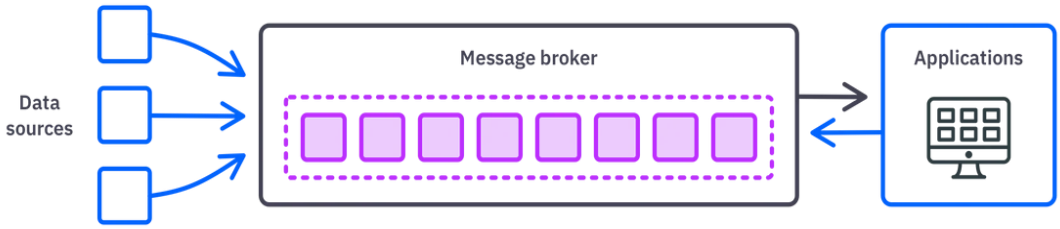
1.1.4 Streamovanie dát

Streamovanie dát (Data Streaming) je proces neustáleho prenosu údajov v reálnom čase alebo blízkom reálnemu času. Streamovanie dát sa líši od tradičného spracovania údajov tým, že nepracuje s uloženými údajmi, ale s „prúdmi“ údajov, ktoré kontinuálne prichádzajú do systému. Tento typ spracovania je výhodný, keď je potrebné analyzovať veľké objemy dát okamžite, bez nutnosti čakania na ich úplné spracovanie. Organizácie majú tisíce zdrojov údajov, ktoré súčasne prenášajú správy, záznamy alebo údaje, ktorých veľkosť sa môže pohybovať od niekoľkých bajtov až po niekoľko megabajtov (Amazon, 2024).

Ďalším plusom je flexibilita a škálovateľnosť. Systémy ako Apache Kafka alebo Apache Flink umožňujú postupné rozširovanie kapacity podľa rastúceho množstva dát, čo znamená, že streamovanie sa ľahko prispôsobí rastúcim nárokom na výkon.

Jedným z hlavných nedostatkov je komplexnosť implementácie a správy takéhoto systému. Ďalšou nevýhodou sú vysoké nároky na infraštruktúru. Spracovanie údajov v reálnom čase kladie obrovské nároky na výpočtový výkon a úložisko, čo môže viesť k vysokým nákladom na prevádzku, najmä pri škálovaní systému. Organizácie často musia investovať do výkonných serverov alebo cloudových riešení, ktoré umožňujú rýchle a stabilné spracovanie veľkých objemov dát.

Ako funguje streamovanie dát?



Obrázok 2: Fungovanie streamovania dát
Zdroj: Rosam, 2021

- Jeden alebo viacero zdrojov údajov vytvára nepretržité toky údajov,
- sprostredkovateľ správ, ako napríklad Apache Kafka, zoskupuje tieto prúdy do tém, čím zabezpečuje konzistentné usporiadanie. Kafka tiež používa replikáciu a môže distribuovať údaje na viacero serverov, aby bola zaistená odolnosť voči chybám,
- aplikácie spotrebúvajú údaje z týchto tém, spracúvajú ich a podľa toho na nich konajú, výsledky spracovaných údajov môžu byť ďalej prenášané späť do sprostredkovateľa správ (Rosam, 2021).

1.1.5 Senzorové siete a IoT

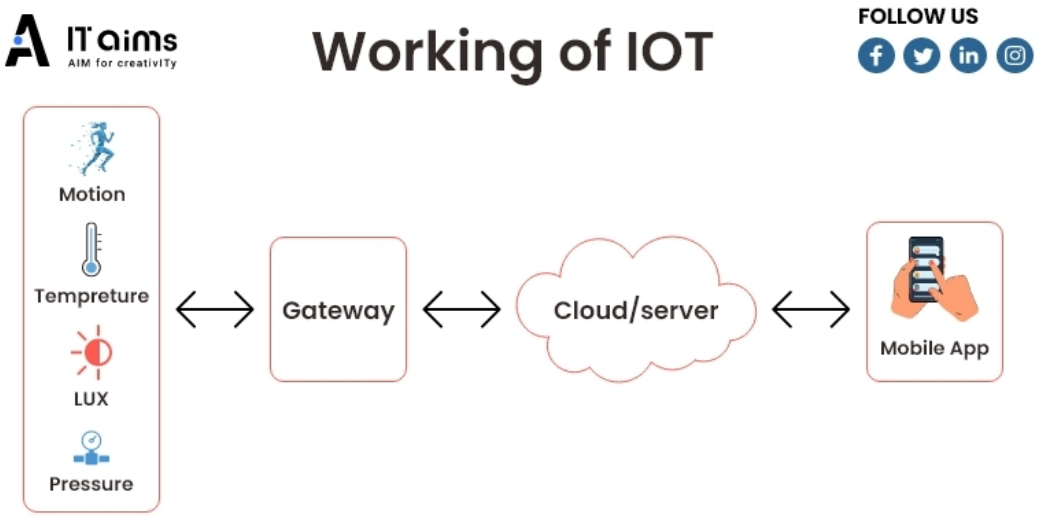
Senzorové siete sú sústavy senzorov rozmiestnených v priestore, ktoré sú navzájom prepojené a prenášajú údaje do centrálnej jednotky alebo databázy. Tieto siete môžu pozostávať z rôznych typov senzorov, ktoré merajú parametre ako teplota, vlhkosť, tlak, pohyb, svetlo či znečistenie ovzdušia. Senzorové siete nachádzajú uplatnenie v monitorovacích systémoch, napríklad v poľnohospodárstve na sledovanie podmienok prostredia, alebo v inteligentných budovách, kde senzory monitorujú spotrebu energie a bezpečnostné podmienky.

Internet vecí (Internet of Things, IoT) rozširuje koncept senzorových sietí tým, že umožňuje, aby zariadenia nielen zbierali údaje, ale aj komunikovali medzi sebou a s centrálou prostredníctvom internetu. IoT umožňuje zariadeniam zdieľať údaje v reálnom

čase a umožňuje prístup k týmto dátam prakticky kdekoľvek, čím sa otvárajú možnosti pre rôzne aplikácie. V IoT systémoch môžu zariadenia komunikovať a reagovať na údaje, čo umožňuje napríklad automatické riadenie výrobných liniek v priemysle alebo inteligentné riadenie dopravy.

Zber údajov IoT je proces používania senzorov na sledovanie stavu fyzických vecí. Zariadenia a technológie pripojené cez IoT dokážu monitorovať a merať dáta v reálnom čase. Údaje sa prenášajú, ukladajú a možno ich kedykoľvek obnoviť.

Výhodou IoT je jeho široká škálovateľnosť a flexibilita, ktorá umožňuje pripojenie tisícov zariadení, ktoré môžu byť riadené na diaľku. IoT platformy, ako sú Azure IoT Hub, AWS IoT Core alebo Google IoT, ponúkajú infraštruktúru pre zber, analýzu a správu dát z veľkého počtu zariadení, čo uľahčuje nasadenie a správu týchto systémov.



Obrázok 3: Fungovanie internetu vecí (IoT)
Zdroj: ITAIIMS, 2023

Zariadenia internetu vecí majú v sebe zabudované senzory. Tieto senzory sú schopné snímať svoje okolie. Zariadenia uchovávajú informácie v nejakej forme údajov. Tieto zariadenia zahŕňajú spotrebiče, ako sú mobilné telefóny, kávovary, mikrovlnné rúry, gejzíry, požiarne hlásiče, klimatizácie, autá atď (ITAIIMS, 2023).

1.1.6 Crowdsourcing a zber dát od používateľov

Crowdsourcing predstavuje inovatívny spôsob, ako zbierať dáta prostredníctvom aktívneho zapojenia veľkého množstva používateľov. Tento prístup využíva kolektívnu inteligenciu a schopnosti širokého spektra jednotlivcov, aby prispeli informáciami, ktoré sú inak náročné na získanie tradičnými metódami zberu dát. Crowdsourcing je čoraz častejšie

používaný vo výskume a vývoji, pretože umožňuje získať rozsiahly objem dát v relatívne krátkom čase a s nižšími nákladmi, než by to bolo možné pri konvenčných prístupoch.

Zber dát od používateľov môže prebiehať rôznymi formami, napríklad prostredníctvom aplikácií, webových formulárov, senzorov v mobilných zariadeniach alebo prostredníctvom sociálnych sietí. Používatelia poskytujú údaje dobrovoľne, často za výmenu za konkrétne výhody, ako napríklad prístup k službám, odmeny alebo zlepšenie personalizácie produktov a služieb. Príkladom crowdsourcingového prístupu môže byť aplikácia, ktorá monitoruje dopravnú situáciu a kde vodiči prispievajú informáciami o premávke v reálnom čase, čo umožňuje vytvoriť presný obraz o aktuálnej situácii na cestách.

Výhody crowdsourcingu spočívajú predovšetkým v jeho flexibilite a širokej dostupnosti. Vďaka veľkému počtu zapojených používateľov môžu byť získané dáta veľmi podrobné a reprezentatívne. Tento prístup sa úspešne využíva v oblastiach ako environmentálny monitoring, predikcia zdravotných trendov, alebo dokonca vo vedeckých výskumoch, kde jednotlivci môžu prispievať k spracovaniu veľkých objemov dát.

Napriek týmto prínosom má crowdsourcing aj svoje slabé stránky. Jedným z hlavných problémov je variabilita v kvalite získaných údajov, keďže prispievatelia majú rozdielne úrovne odbornosti, skúseností a motivácie. To môže viesť k tomu, že niektoré informácie budú nepresné alebo neúplné. Navyše, pri absencii kvalitného systému kontroly môže byť náročné odhaliť chybné alebo úmyselne skreslené dáta. Z tohto dôvodu je vhodné implementovať verifikačné mechanizmy, ako napríklad krížovú validáciu dát, zber viacnásobných odpovedí od rôznych používateľov, alebo využitie algoritmických metód na detekciu nezrovnalostí (Estellés-Arolas & González, 2012).

1.1.7 Open Data Portály a dátové banky

Open Data portály a dátové banky predstavujú významný zdroj informácií, ktoré poskytujú prístup k voľne dostupným údajom od štátnych inštitúcií, neziskových organizácií a výskumných inštitútov. Tieto údaje sú zvyčajne štruktúrované v súlade s medzinárodnými štandardmi, čo umožňuje ich jednoduchú analýzu a integráciu do rôznych projektov (Janssen, 2012). Medzi najvýznamnejšie nástroje pre prístup k otvoreným dátam patria:

- **Data.gov**: Americký portál poskytujúci široké spektrum dátových súborov, pokrývajúcich oblasti od dopravy až po environmentálne dáta.
- **World Bank Data**: Platforma zameraná na poskytovanie údajov o svetových ekonomikách, ktoré sú využiteľné na analýzu makroekonomických a sociálnych trendov.

- **Eurostat**: Európska databáza štatistických údajov pokrývajúca ekonomické, demografické a environmentálne dáta pre členské štáty EÚ.

1.1.8 Robotické procesy

Robotické procesy (Robotic Process Automation, RPA) predstavujú technológiu určenú na automatizáciu rutinných manuálnych úloh prostredníctvom softvérových robotov. Tieto nástroje dokážu samostatne vykonávať opakujúce sa činnosti, ako napríklad získavanie údajov z webových stránok, informačných systémov či elektronických dokumentov. Zavedením RPA sa znižuje časová náročnosť spracovania údajov a zároveň sa minimalizuje riziko chýb spôsobených manuálnym zásahom človeka. V dôsledku toho sa zvyšuje objem spracovaných údajov a zlepšuje sa konzistentnosť výstupov. Napríklad, štúdia z roku 2024 ukázala, že implementácia RPA v prostredí Agile a DevOps prispieva k zrýchleniu dodacích cyklov a zlepšeniu celkovej kvality softvéru (Manukonda, 2024) .

RPA je obzvlášť užitočná v situáciách, keď nie je dostupné API alebo keď sú procesy zberu údajov zložité a vyžadujú manuálny zásah. Softvérové nástroje ako UiPath, Automation Anywhere alebo Blue Prism umožňujú vytvárať roboty, ktoré automatizovane prehľadávajú webové stránky, extrahujú relevantné informácie a ukladajú ich do preddefinovaného formátu, napríklad CSV, JSON alebo databázovej štruktúry. Tieto nástroje sú schopné interagovať s používateľským rozhraním aplikácií, čo umožňuje automatizáciu aj v prípade, že aplikácie nemajú dostupné API (Kumar & Deepak, 2024).

Výhodou RPA je predovšetkým schopnosť automatizovať úlohy, ktoré by inak vyžadovali veľa času a boli by náchylné na chyby. Navyše umožňuje rýchlu integráciu s existujúcimi systémami bez nutnosti zložitých zmien v infraštruktúre. Na druhej strane, implementácia RPA môže byť technicky náročná a počiatočné náklady na jej zavedenie sú relatívne vysoké.

1.1.9 Zber a ukladanie údajov v architektúre Big Data

S rastúcim objemom dát generovaných v reálnom čase z rôznych zdrojov, ako sú senzorové siete, API rozhrania, sociálne siete či webové portály, sa čoraz viac využívajú technológie spojené s architektúrou Big Data. Táto architektúra je navrhnutá tak, aby dokázala zvládať veľké objemy, rôznorodosť a rýchlosť prichádzajúcich údajov (tzv. „5V“: velocity, volume, value, variety and veracity).

V kontexte zberu a spracovania údajov v reálnom čase sa Big Data systémy zameriavajú najmä na vysoko priepustný príjem dát, ich predbežné triedenie, dočasné

ukladanie a prípravu na ďalšie analytické spracovanie. Na tento účel sa využívajú technológie ako:

- **Apache Kafka** je distribuovaná platforma určená na spracovanie a prenos dátových tokov. Umožňuje vysokopriepustné prijímanie a sprostredkovanie údajov medzi rôznymi systémami v reálnom čase. V praxi to znamená, že Kafka dokáže spracovať milióny záznamov za sekundu, pričom zabezpečuje ich spoľahlivé doručenie, archiváciu a možnosť opätovného prečítania. Využíva sa ako medzičlánok medzi zdrojmi údajov (napr. senzory, internetové služby, logy systémov) a cieľovými systémami (napr. databázy, analytické nástroje), čím podporuje oddelenie komponentov a zvyšuje flexibilitu celého systému.
- **Apache NiFi** – nástroj na vizuálne riadenie tokov dát medzi rôznymi systémami,
- Apache Hadoop (Hadoop Distributed File System, HDFS) – je distribuovaný súborový systém, ktorý umožňuje ukladanie veľmi veľkých objemov údajov rozdelených medzi viacero serverov. Každý súbor je rozdelený na menšie časti (bloky), ktoré sú následne uložené a zálohované na viacerých miestach v rámci výpočtového klastra.
- **Apache Spark** – predstavuje moderný výpočtový nástroj na spracovanie údajov vo veľkom rozsahu. Podporuje ako dávkové spracovanie (spracovanie veľkých súborov údajov v určitých intervaloch), tak aj spracovanie údajov v reálnom čase (Apache Kafka Documentation, 2025).
- **NoSQL databázy** – ako napríklad **Cassandra** a **MongoDB**, sú moderné databázové systémy navrhnuté pre prácu s veľkými a rôznorodými údajmi. Na rozdiel od tradičných relačných databáz, ktoré používajú pevne definované tabuľky, NoSQL databázy umožňujú flexibilné ukladanie údajov vo forme dokumentov, párov kľúč-hodnota alebo grafových štruktúr (Rosam, 2021).

Zber údajov v Big Data architektúre je často realizovaný cez tzv. ingestion vrstvy, ktoré zabezpečujú efektívne prijímanie údajov z viacerých zdrojov súčasne. Tieto údaje sú následne ukladané do vyrovnávacích pamätí alebo priamo do distribuovaných úložísk, kde môžu byť ďalej analyzované pomocou paralelných výpočtových mechanizmov.

Architektúra je navrhnutá s dôrazom na škálovateľnosť, odolnosť voči výpadkom a nízku latenciu. Využitie týchto nástrojov umožňuje nielen ukladanie veľkých dátových objemov, ale aj ich spracovanie v reálnom čase, čo je zásadné pri monitorovaní vývoja kurzov, trendov alebo udalostí s časovou citlivosťou (Kune et al., 2015).

1.2 Analýza časových radov

Kapitola sa venuje opisu základných charakteristík časových radov, ich typológii a špecifikám finančných časových radov, poskytuje teoretický základ, ktorý je kľúčový pre neskoršie pochopenie predikčných modelov diskutovaných v nasledujúcej kapitole 1.3.

1.2.1 Základné zložky časového radu

Časový rad predstavuje postupnosť údajov usporiadaných podľa diskrétneho časového sledu, pričom sa bežne vyskytuje v reálnych aplikáciách, ako je hodnotenie finančných rizík, udržateľnosť energetiky či predpoveď počasia (Hamilton, 1994).

Základné zložky časového radu sú hlavné faktory, ktoré ovplyvňujú vývoj hodnoty sledovaného ukazovateľa v čase. Časový rad sa spravidla skladá z týchto štyroch hlavných zložiek:

- **Trend**

Trend je dlhodobý smer pohybu v časovom rade a môže byť rastúci, klesajúci alebo stabilný. Chatfield (2004) uvádza, že trend v časovom rade sa môže prejavovať ako stabilný nárast alebo pokles hodnôt v priebehu času, často súvisiaci so širšími makroekonomickými trendmi.

- **Sezónnosť**

Sezónnosť označuje pravidelné, periodické zmeny, ktoré sa objavujú v časovom rade na základe určitých časových intervalov, napríklad ročných období, mesiacov či týždňov.

- **Cyklickosť**

Reprezentuje dlhodobejšie zmeny spojené s hospodárskymi cyklami. Tieto cykly môžu trvať niekoľko rokov a súvisia napríklad s hospodárskymi krízami, expanziami alebo inými ekonomickými faktormi.

- **Náhodné zložky a šum**

Každý časový rad obsahuje náhodné zložky alebo šum, ktoré sú nepredvídateľné a ovplyvňujú presnosť predikcie. Šum môže byť spôsobený neočakávanými udalosťami, ktoré nemajú vzor ani logiku (napr. prírodné katastrofy, politické prevraty). Šum sťažuje predikciu, pretože znižuje zrozumiteľnosť štruktúry údajov a pridáva ďalší rozmer nepredvídateľnosti.

Tieto zložky môžeme zapísať do dvoch modelov — **aditívneho a multiplikatívneho modelu.**

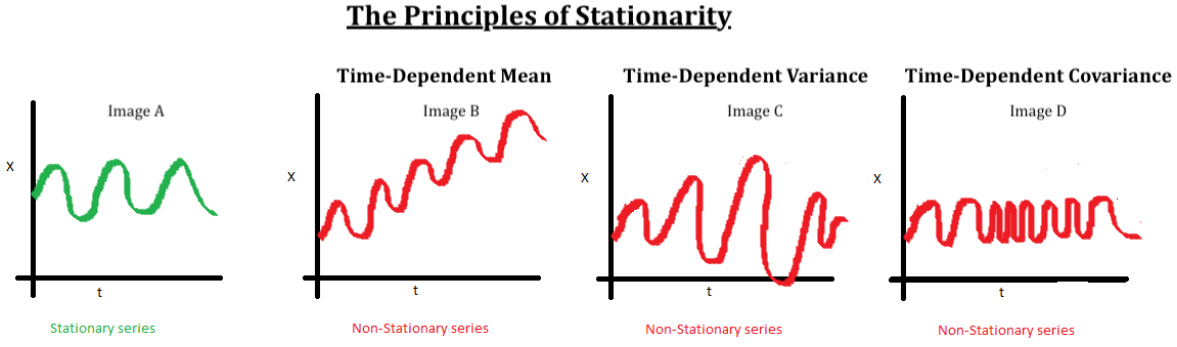
1.2.2 Charakteristiky časového radu

Autokorelácia je charakteristikou časového radu, ktorá popisuje mieru závislosti medzi hodnotami v rôznych časových bodoch v rámci jedného radu. Autokorelácia pomáha identifikovať, či a do akej miery sú predchádzajúce hodnoty v rade relevantné pre predpoveď budúcich hodnôt. Box a Jenkins (1976) tvrdia, že autokorelačné vzorce sú kľúčové pri identifikácii sezónnych alebo cyklických efektov, ktoré môžu byť využité pri vytváraní presnejších predikčných modelov.

Volatilita označuje mieru variability alebo kolísania hodnôt v časovom rade. Volatilita sa mení v čase, podmienený rozptyl nie je konštantný, t.j. časový rad je podmienene heteroskedastický (Rublíková & Prihodová, 2008). To znamená, že rozptyl hodnôt časového radu nie je rovnaký počas celého obdobia, ale závisí od predchádzajúcich hodnôt alebo iných faktorov. Tento jav je bežný vo finančných časových radoch, ako sú výmenné kurzy alebo ceny akcií, kde sa obdobia vysokej volatility striedajú s obdobiami nízkej volatility.

Prvý, kto sa začal zaoberať skúmaním volatility v časových radoch, bol Robert F. Engle, ktorý vytvoril autoregresný podmieniene heteroskedastický model ARCH (AutoRegressive Conditional Heteroskedasticity), ktorý sa stal základom pre ďalšie lineárne i nelineárne modely podmieneného rozptylu (Rublíková & Prihodová, 2008).

Stacionárita. Časový rad je stacionárny, ak jeho správanie v rôznych časových intervaloch zostáva konzistentné a nevykazuje systematické zmeny v čase. Stacionárita je dôležitá pri modelovaní a predikcii, pretože mnoho štatistických a ekonometrických modelov predpokladá, že údaje majú konštantnú štruktúru.



Obrázok 4: Stacionárita časového radu
Zdroj: Mitrani, 2020

1. Priemer časového radu by nemal byť funkciou času. Červený graf (Obr. 4 B) nie je stacionárny, pretože priemer narastá v čase.

2. Rozptyl časového radu by nemal byť funkciou času. Táto vlastnosť sa nazýva homoskedasticita.

3. Kovariancia medzi i-tým a (i + m)-tým členom by nemala byť funkciou času. Na nasledujúcom grafe si môžete všimnúť, že vzdialenosti medzi dátami sa s časom menia (v červenom grafe sa stávajú bližšie). Preto kovariancia nie je konštantná v čase pre „červený časový rad“ na Obr. 4 D. Stacionárne časové rady majú konštantný priemer, rozptyl a kovarianciu v priebehu času, čo znamená, že ich štatistické vlastnosti sú časovo invariantné (Mitrani, 2020).

Stacionárnosť je kľúčovým predpokladom mnohých modelov časových radov, ako sú ARIMA alebo GARCH, ktoré budú podrobnejšie analyzované v nasledujúcich kapitolách.

Normalita, Finančné časové rady často nevykazujú normalitu rozdelenia svojich hodnôt. Napriek predpokladu, že výnosy aktív alebo zmeny kurzov nasledujú normálne rozdelenie, empirické údaje často vykazujú:

Fat tails (hrubé chvosty): Pravdepodobnosť extrémnych hodnôt je väčšia, ako predpokladá normálne rozdelenie.

Asymetriu: Rozdelenie nemusí byť symetrické okolo strednej hodnoty.

Linearita. Finančné časové rady sú často ne-lineárne, čo znamená, že vzťahy medzi premennými nie sú jednoducho vyjadriteľné lineárnymi rovnicami. Typické znaky nelinearity zahŕňajú:

Závislosť volatility na čase: Volatilita má tendenciu zoskupovať sa do období vysokej alebo nízkej variability (tzv. "volatility clustering").

Náhodné skoky: Ceny aktív môžu vykazovať náhle zmeny, ktoré nie je možné predpovedať.

Asymetrická odpoveď na informácie: Trhy často reagujú odlišne na pozitívne a negatívne správy, čo pridáva k zložitosti modelovania (Rublíková &Príhodová, 2008).

1.2.3 Druhy časových radov

Z odlišného charakteru údajov zoradených v časových radoch vyplýva potreba ich rozdelenia podľa viacerých hľadísk. Podľa frekvencie, s akou sú údaje zbierané, rozlišujeme dlhodobé a krátkodobé časové rady:

1. **Dlhodobé časové rady** vznikajú v prípade zisťovania hodnôt ukazovateľa za ročné alebo dlhšie obdobie.
2. **Krátkodobé časové rady** sú zostavené z hodnôt zistených za obdobia kratšie ako rok, teda štvrťroky, mesiace, týždne. Časové rady denných hodnôt a hodnôt, ktoré sa zisťujú v časových intervaloch (hodiny, minúty či sekundy), sa pre vysokú frekvenciu zisťovania

nazývajú aj **vysokofrekvenčné časové rady**. Napríklad denné hodnoty menového kurzu SKK/USD vyhlásené Národnou bankou Slovenska k 18.00 hod. Od 1.5.2004 do 1.5.2005 a poď (Rublíková, 2007).

Časové rady môžu byť klasifikované podľa toho, či údaje reprezentujú celkové hodnoty za určitý interval alebo stav v konkrétnom okamihu:

1. **Okamihové** časové rady sú zostavené z hodnôt, ktoré sa vzťahujú na určitý okamih.
2. **Intervalové** zahŕňajú údaje, ktoré vyjadrujú priemernú alebo kumulatívnu hodnotu za určitý časový interval.

Časové rady môžeme klasifikovať aj podľa druhu sledovaných ukazovateľov, pričom rozlišujeme časové rady primárnych a sekundárnych ukazovateľov:

1. Časové rady primárnych ukazovateľov obsahujú údaje, ktoré sú priamo merané alebo pozorované. Slúžia ako východiskový bod analýzy.

2. Časové rady sekundárnych ukazovateľov sú odvodené od primárnych ukazovateľov pomocou matematického alebo štatistického spracovania. Umožňujú lepšie pochopiť zložitejšie súvislosti v dátach.

Podľa počtu skúmaných charakteristík môžeme časové rady rozdeliť na jednorozmerné a mnohorozmerné:

1. Jednorozmerné časové rady sa zameriavajú na sledovanie vývoja jednej konkrétnej premennej v čase.

2. Mnohorozmerné časové rady sledujú vývoj viacerých premenných súčasne, pričom tieto premenné môžu byť navzájom prepojené alebo korelované (Rublíková, 2007).

Tabuľka 1: Druhy časových radov
Zdroj: vlastné spracovanie

Príznak klasifikácie	Druhy časových radov
Podľa periodicity sledovania	Dlhodobé Krátkodobé
Podľa rozhodného časového hľadiska zisťovania	Intervalové Okamihové
Podľa kvalitatívnych vlastností skúmaného javu	Absolútne Relatívne Priemerné hodnoty
Podľa druhu sledovaných ukazovateľov	Časové rady primárnych ukazovateľov Časové rady sekundárnych ukazovateľov
Podľa počtu skúmaných charakteristík	Jednorozmerné Mnohorozmerné

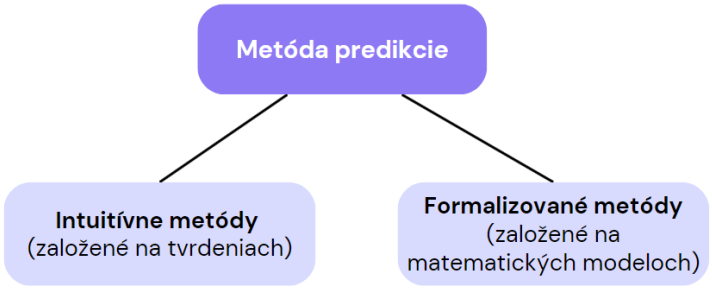
1.3 Modely predikcie časových radov

V oblasti predikcie časových radov sa často stretávame s množstvom rôznych modelov a metód, ktoré autori vo svojich prácach spravidla iba uvádzajú, bez hlbšieho zámeru ich systematicky klasifikovať. Avšak s rastúcim množstvom dostupných techník a ich špecifickými aplikáciami je čoraz ťažšie sa v tejto problematike orientovať. Preto považujeme za nevyhnutné vytvoriť prehľadnú a logickú klasifikáciu metód a modelov časových radov.

Metóda predikcie je systematický postup alebo súbor krokov používaných na predpovedanie budúceho vývoja javov a procesov. Cieľom je znížiť mieru neistoty budúceho vývoja stanovením kvalitatívnych a kvantitatívnych parametrov budúcich udalostí a termínov ich vzniku.

Model predikcie je matematický alebo štatistický nástroj, ktorý opisuje vzťahy medzi rôznymi premennými a umožňuje predpovedať budúce hodnoty na základe historických údajov.

Ak sa na problematiku pozrieme podrobnejšie, rýchlo zistíme, že pojem „metóda predikcie“ je oveľa širší ako pojem „model“. Z tohto dôvodu môžeme metódy na prvom kroku klasifikácie deliť na dve hlavné skupiny: intuitívne a formalizované.



Obrázok 5: Metódy predikcie
Zdroj: vlastné spracovanie

1. Intuitívne metódy, ktoré sú často nazývané aj kvalitatívne, sú založené na tvrdeniach a odhadoch odborníkov. Tieto metódy sa dnes často využívajú v oblastiach, ako je marketing, ekonomika a politika, pretože systém, ktorého správanie je potrebné predvídať, je buď veľmi zložitý a matematicky neopísateľný, alebo naopak veľmi jednoduchý a matematické opisovanie nie je potrebné. Medzi najčastejšie používané intuitívne metódy patria:

- Expertné hodnotenie – odhad budúceho vývoja na základe skúseností odborníkov,
- Metóda Delphi – štruktúrovaný proces získavania kolektívneho názoru expertov,

- Panelový konsenzus – získanie predikcie na základe dohody viacerých odborníkov.

2. Formalizované metódy predikcie (tiež označované ako **kvantitatívne** metódy) sú založené na matematických a štatistických modeloch. Tieto metódy definujú matematické vzťahy, ktoré umožňujú vypočítať budúce hodnoty procesov, čím poskytujú možnosť predikcie (Čučueva, 2012).

Klasifikácia modelov predikcie

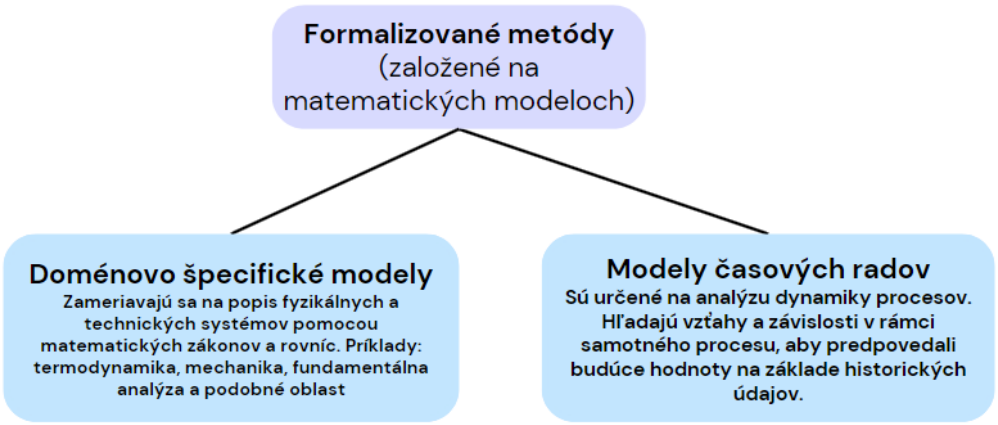
Na základe vyššie uvedenej diskusie môžeme pristúpiť ku klasifikácii modelov predikcie. Na prvom kroku je vhodné rozdeliť modely na dve hlavné kategórie:

1. Doménovo špecifické modely:

Tieto modely sú navrhnuté na základe zákonitostí a vlastností špecifických pre konkrétnu oblasť (predmetový rámec), napríklad termodynamiku, mechaniku, ekonómiu, medicínu alebo inžinierstvo. Pri ich tvorbe sa využívajú odborné znalosti z danej oblasti, ktoré umožňujú lepšie pochopiť vnútorné vzťahy a procesy. Tieto modely sú veľmi presné pre konkrétnu oblasť, avšak ich použitie je spravidla obmedzené na daný kontext.

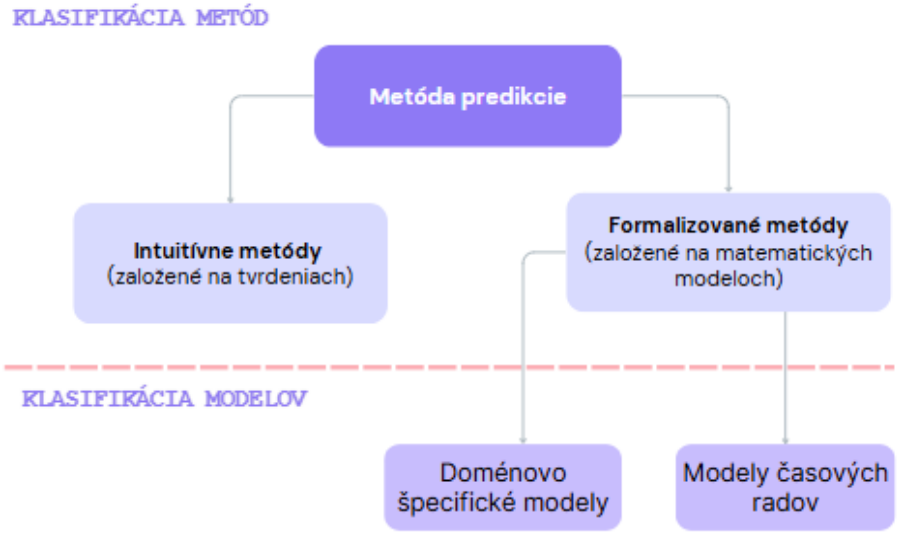
2. Modely časových radov:

Tieto modely sú založené na analýze vnútorných vzťahov v samotnom procese a na identifikácii závislostí medzi minulými a budúcimi hodnotami. Sú univerzálnejšie, pretože ich použitie nie je striktne viazané na konkrétnu oblasť. Takéto modely sú vhodné pre aplikácie, kde je možné využiť historické dáta na predikciu budúcich trendov (Čučueva, 2012).



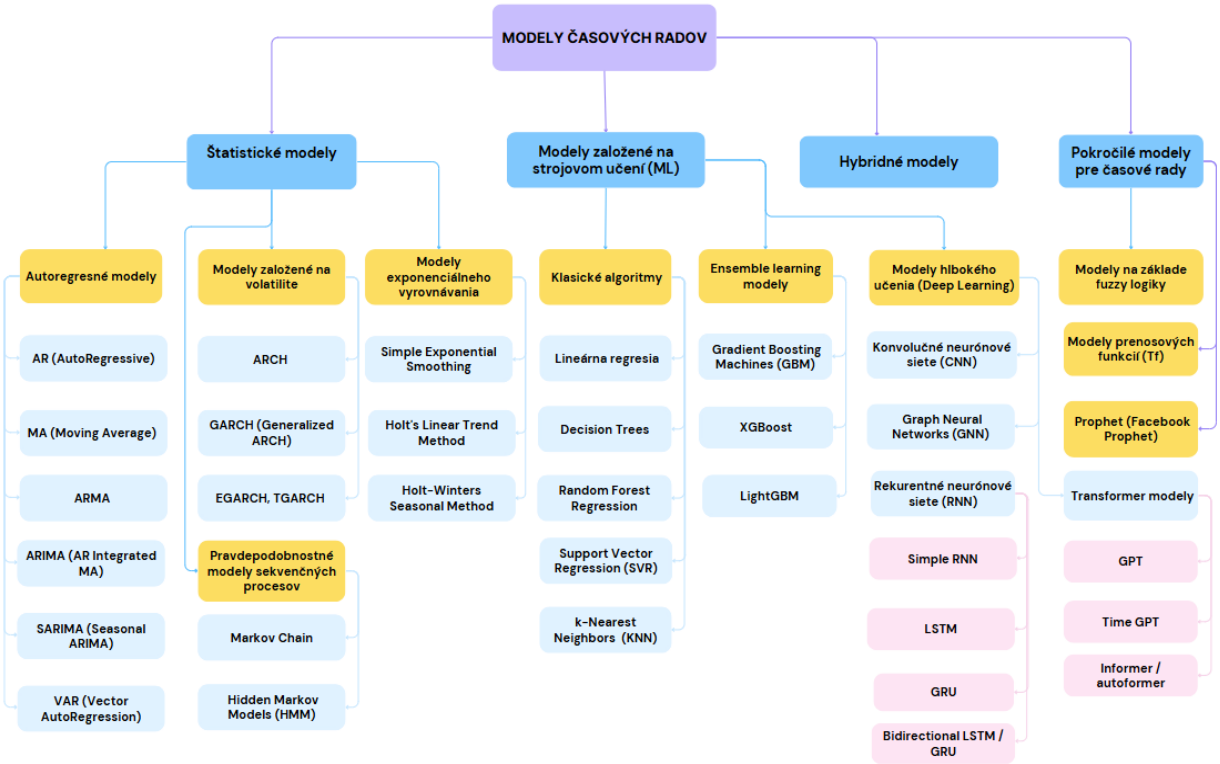
Obrázok 6: Formalizované metódy
Zdroj: vlastné spracovanie

Týmto spôsobom sme vytvorili základnú klasifikáciu metód a modelov predikcie (Obr. 7), ktorá umožňuje lepšie porozumenie ich štruktúre a vzájomným vzťahom.



Obrázok 7: Klasifikácia metód a modelov predikcie
Zdroj: vlastné spracovanie

Modely časových radov možno rozdeliť na viacero kategórií podľa princípu fungovania a prístupu k spracovaniu dát. Základné delenie zahŕňa štatistické modely a modely strojového učenia. Prepojením viacerých prístupov vznikajú hybridné modely, zatiaľ čo pokročilé modely ponúkajú riešenia pre špecifické prípady predikcie časových radov.



Obrázok 8: Modely časových radov
Zdroj: vlastné spracovanie

V nasledujúcich podkapitolách sú predstavené vybrané modely predikcie časových radov, ktoré sa v súčasnosti využívajú v odbornej praxi a výskume.

1.3.1 Štatistické modely

Ide o klasické prístupy časových radov, často založené na predpoklade linearity a stacionarity dát.

Autoregresné modely sú štatistické modely používané na analýzu časových radov, kde sú aktuálne hodnoty predpovedané na základe lineárnej kombinácie minulých hodnôt (Hyndman, & Athanasopoulos, 2018).

Jedným z predpokladov autoregresných modelov je **stacionarita**, ktorú sme bližšie vysvetlili v kapitole 1.2.2. Ďalším predpokladom je — **ergodicita**. To znamená, že štatistické vlastnosti časových radov, ako je priemer a rozptyl , by sa nemali v priebehu času meniť. Aj keď ergodicita nie je striktne požiadavkou pre použitie ARIMA modelov, jej prítomnosť uľahčuje analýzu a interpretáciu dlhodobého správania procesu, pretože umožňuje získať reprezentatívne štatistiky z jednej časti radu počas dlhého obdobia (Whitenack & Selvaraj, 2019).

Modely ARIMA (Autoregressive Integrated Moving Average) sú často využívané na predikciu časových radov, ktoré sú stacionárne alebo nestacionárne, ale môžu byť transformované na stacionárne aplikovaním diferenciácie.

Rozšírenie ARIMA modelov, **SARIMA** (Seasonal ARIMA), zahŕňa sezónne komponenty, čím umožňuje modelovať opakujúce sa sezónne vzorce v časových radoch.

Vektorové autoregresné modely (VAR) sú jednou z najpopulárnejších metód na analýzu a predikciu viacrozmerných časových radov. Tento typ modelu bol predstavený Christopherom Simsom v roku 1980 ako alternatíva k tradičným ekonometrickým modelom, ktoré si vyžadovali silné teoretické obmedzenia. Sims (1980) argumentoval, že v makroekonomických a finančných časových radoch často nepoznáme presné štrukturálne vzťahy, a preto je vhodné nechať dáta určovať vzťahy medzi premennými na základe ich historických hodnôt. VAR modely sú široko používané pri analýze makroekonomických dát, finančných trhov a iných oblastí, kde je potrebné modelovať vzájomné závislosti medzi viacerými premennými.

Modely ARCH (Autoregressive Conditional Heteroskedasticity) sú špecifickým typom autoregresných modelov, ktoré sú navrhnuté na analýzu a predikciu časových radov s premenlivou (heteroskedastickou) volatilitou. Tieto modely boli predstavené Robertom Englom (1982), ktorý za svoju prácu získal Nobelovu cenu za ekonómiu. Ich základným

princípom je, že podmienená variabilita (resp. volatilita) časového radu sa mení v čase a závisí od minulých chýb modelu.

GARCH modely (Generalized Autoregressive Conditional Heteroskedasticity) rozširujú prístup ARCH modelov tým, že pri odhade volatility zohľadňujú nielen predchádzajúce chyby, ale aj predchádzajúce hodnoty samotnej volatility. Vďaka tomu dokážu lepšie vystihnúť meniacu sa volatilitu časových radov, najmä v prípadoch, kde sa striedajú pokojné a turbulentné obdobia.

Preto je potrebné pri praktickom využití týchto modelov systematicky testovať ich predpoklady a, ak je to potrebné, uvažovať o ich rozšíreniach, ako sú EGARCH alebo GJR-GARCH.

Exponenciálne vyrovnávanie (ES) — ide o rodinu metód (Simple, Double, Triple/Holt-Winters), ktoré vytvárajú predikciu ako exponenciálne vážený priemer minulých hodnôt. Exponenciálne vyrovnávanie navrhol v roku 1956 Robert Goodel Brown, neskôr ho v roku 1957 rozšíril pán Charles Holt. Brownovo exponenciálne vyrovnávanie je možné použiť na nesezónne dáta. **Holt-Winters** metóda napríklad uvažuje komponenty úrovne, trendu a sezónnosti, ktoré priebežne aktualizuje na základe nových dát. Tieto metódy sú menej formálne štatistické a viac heuristické, ale často poskytujú dobré výsledky najmä pre **hladké** časové rady.

1.3.2 Klasické algoritmy strojového učenia

Modely strojového učenia nepredpisujú pevný štatistický vzťah, ale učia sa ho z dát. Medzi klasické modely strojového učenia patria rôzne regresné metódy.

Lineárna regresia je jedným z najjednoduchších prístupov pre analýzu a predikciu časových radov. Tento nástroj predpokladá existenciu lineárneho vzťahu medzi závislou premennou a nezávislými premennými (Montgomery, Peck & Vining, 2012).

Lineárna regresia umožňuje vytvoriť model, ktorý sa používa na vytvorenie rovnice, opisujúcej vzťah medzi minulosťou a budúcnosťou časového radu. Tento model je ideálny pre prípad, keď existuje priamy a predvídateľný vzťah medzi hodnotami v časovom rade, čo je vhodné pre jednoduché a dobre definované systémy. V reálnom svete však časové rady často vykazujú komplexné a nelineárne správanie, ktoré môže byť ťažké zachytiť jednoduchou lineárnou regresiou (Hyndman & Athanasopoulos, 2018).

Mnohonásobná lineárna regresia rozširuje tento prístup na viacero nezávislých premenných, umožňujúc analyzovať komplexnejšie vzťahy (James et al., 2021).

Polynomická regresia je vhodná na modelovanie nelineárnych vzťahov medzi premennými, zatiaľ čo logistická regresia sa využíva na predikciu pravdepodobností binárnych výsledkov. Pri práci s veľkým množstvom premenných sú užitočné **Ridge a Lasso regresie**, ktoré pomáhajú znižovať nadmernú komplexnosť modelu a predchádzať preťaženiu modelu (overfitting).

Bayesovská regresia umožňuje zahrnúť predchádzajúce informácie a je vhodná v podmienkach neistoty (Gelman et al., 2014). Automatizované postupy, ako **krokové regresie** (stepwise regression), uľahčujú výber relevantných premenných, zatiaľ čo **kvantilová regresia** analyzuje vzťahy na rôznych úrovniach rozdelenia, čo je užitočné pre analýzu extrémnych hodnôt (Koenker, 2005).

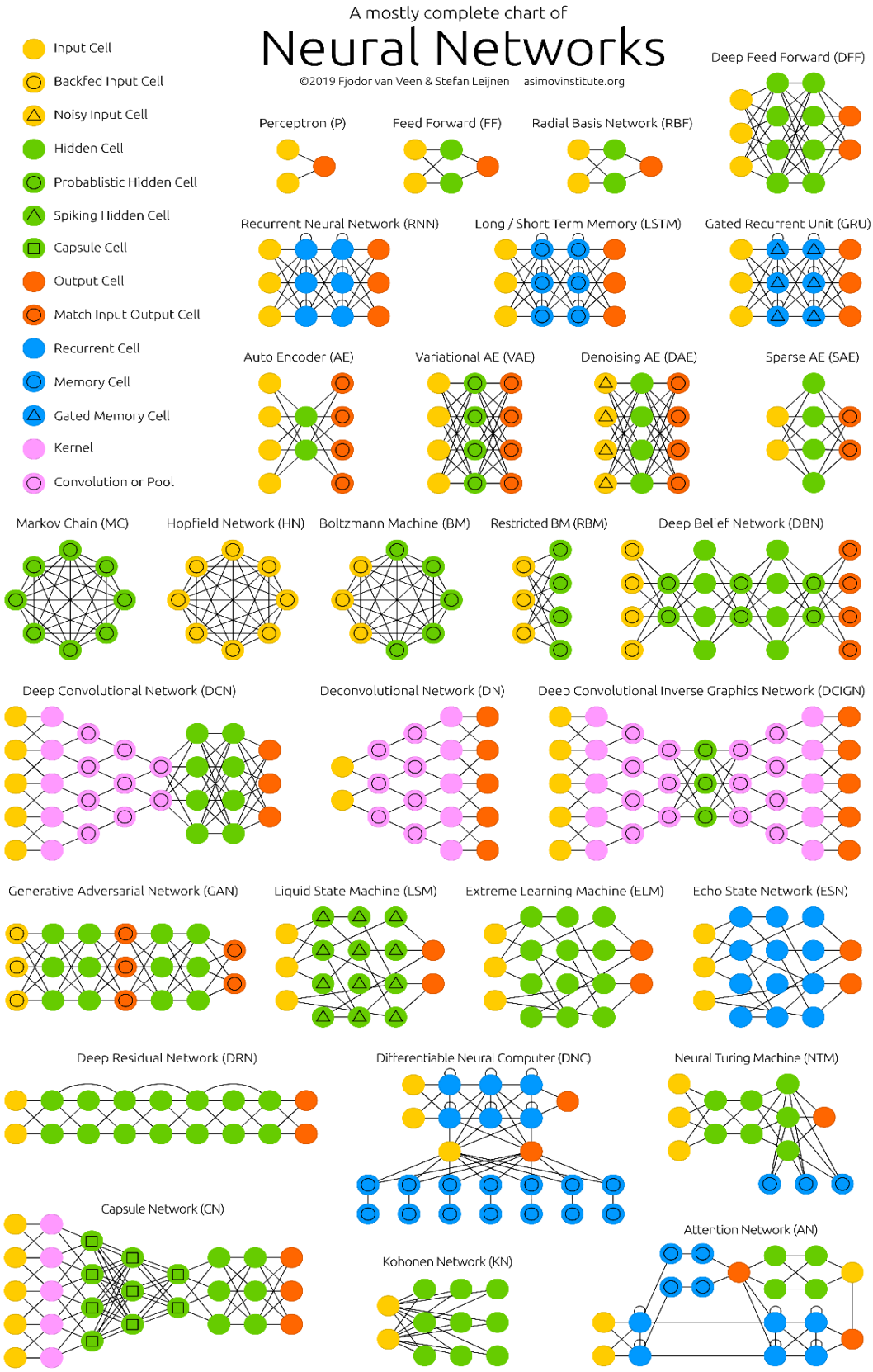
Regresia pomocou metódy podporných vektorov (Support Vector Regression, SVR) — ide o aplikáciu podporných vektorových strojov**,** ktorá sa v oblasti strojového učenia používa predovšetkým na klasifikačné úlohy, no nachádza uplatnenie aj pri regresii. SVR hľadá funkciu (napr. nelineárnu pomocou kernelov), ktorá aproximuje vzťah medzi vstupnými a výstupnými hodnotami s čo najmenšou chybou, pričom toleruje malé chyby. SVR dokáže modelovať aj zložitejšie nelineárne vzťahy a v porovnaní s lineárnymi modelmi (ARIMA) vie často dosiahnuť lepšiu presnosť, najmä ak je časový rad ovplyvnený viacerými faktormi.

Náhodný les (Random Forest, RF) — ansámblový stromový model, ktorý vytvára veľké množstvo rozhodovacích stromov (každý na náhodne zvolenom podvzorku dát a vstupných premenných) a ich výstupy spriemeruje.

Gradient boosting je ďalšia ansámblová technika založená na stromoch, kde stromy sú stavané sekvenčne tak, aby postupne vylepšovali chyby predchádzajúcich stromov. **XGBoost** je populárna implementácia gradientového boostingu, známa efektivitou a vysokou presnosťou v štruktúrovaných dátach. RF/XGBoost môže slúžiť na odhad intervalov alebo pravdepodobnostných scenárov (napr. 70% šanca, že kurz o týždeň prekročí určitú hranicu).

1.3.3 Modely hlbokého učenia — neurónové siete

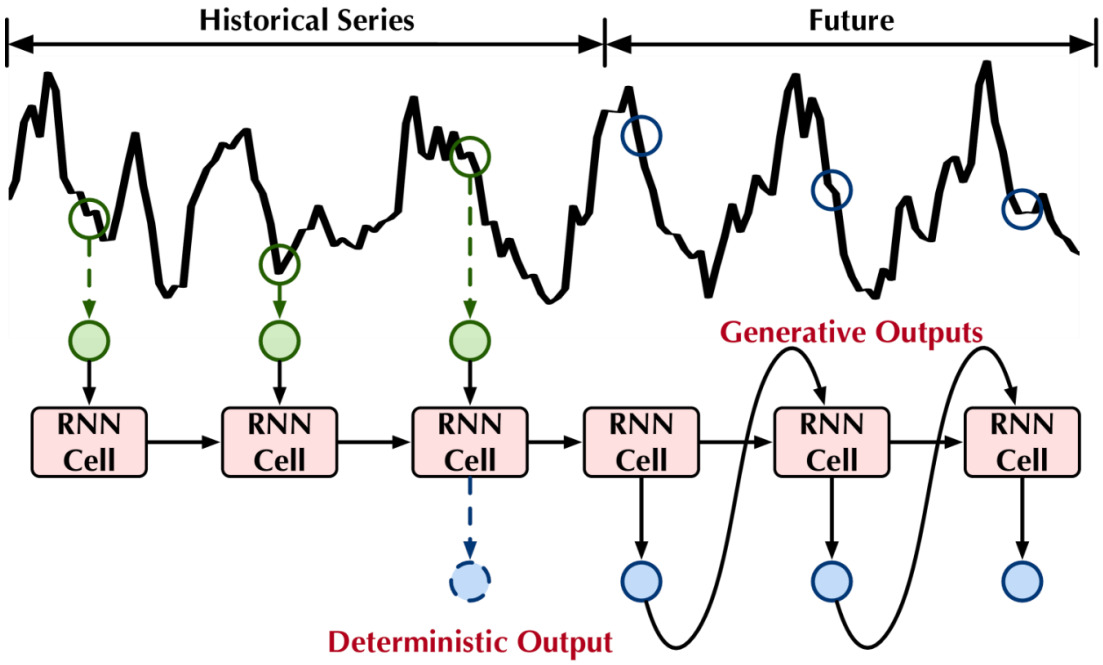
Neurónové siete sú modely inšpirované biologickými neurónovými sieťami, ktoré sa používajú na spracovanie a analýzu dát s cieľom rozpoznávať vzory a vykonávať predikcie. Sú základom hlbokého učenia (deep learning) a patria medzi najvýkonnejšie nástroje v oblasti umelej inteligencie. Existuje mnoho typov neurónových sietí (Obr. 9), pričom každá je určená na špecifické úlohy.



Obrázok 9: Komplexný prehľad neurónových sietí
Zdroj: Van Veen, F. & Leijnen, S. (2019)

Predikcia časových radov vyžaduje špeciálne modely, ktoré dokážu zachytávať závislosti medzi hodnotami v čase. Medzi rôznymi typmi neurónových sietí zohrávajú kľúčovú úlohu pri spracovaní časových radov najmä rekurentné neurónové siete (RNN).

RNN predstavujú špecifický typ architektúry hlbokého učenia, navrhnutý na spracovanie sekvenčných údajov. Pri predikcii časových radov je dôležité zachytávať závislosti medzi jednotlivými časovými krokmi, čo klasické dopredné neurónové siete nedokážu zabezpečiť.



Obrázok 10: Ilustrácia modelu založeného na RNN v úlohe prognózovania
Zdroj: Wang, 2024

Long Short-Term Memory (dlhodobá a krátkodobá pamäť, LSTM) je druh rekurentnej neurónovej siete, ktorá dokáže zachytiť nelineárne a komplexné vzťahy medzi premennými, ktorý navrhli Hochreiter & Schmidhuber (1997).

Tradičné RNN má jeden skrytý stav, ktorý prechádza časom, čo môže sieti sťažiť naučenie sa dlhodobých závislostí. Model LSTM rieši tento problém zavedením pamäťovej bunky, čo je kontajner, ktorý môže uchovávať informácie na dlhšiu dobu.

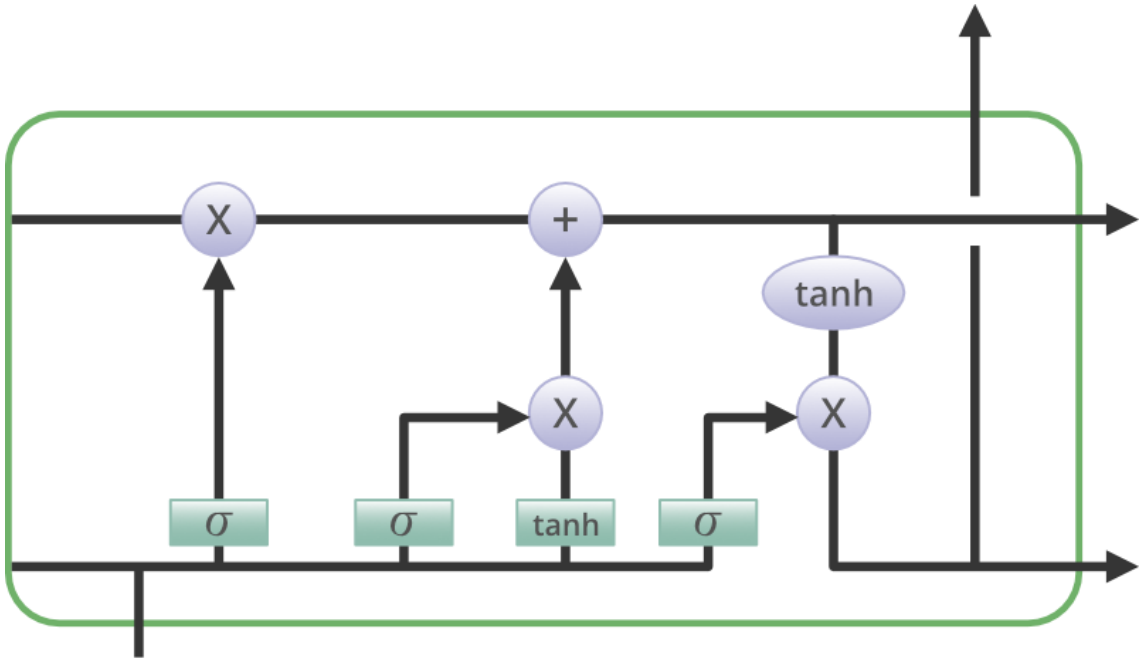
Základný rozdiel medzi architektúrami RNN a LSTM je v tom, že skrytá vrstva LSTM je hradlová jednotka alebo hradlová bunka. Skladá sa zo štyroch vrstiev, ktoré sa navzájom ovplyvňujú tak, aby spolu so stavom bunky produkovali výstup tejto bunky. Tieto dve veci sa potom prenesú na ďalšiu skrytú vrstvu. Na rozdiel od RNN, ktoré majú iba jednu vrstvu neurálnej siete tanh, LSTM obsahujú tri logistické sigmoidné brány a jednu vrstvu tanh. Brány boli zavedené s cieľom obmedziť informácie, ktoré prechádzajú bunkou. Určujú,

ktorú časť informácie bude nasledujúca bunka potrebovať a ktorá časť sa má vyradiť. Výstup je zvyčajne v rozsahu 0-1, kde „0“ znamená „odmietnuť všetko“ a „1“ znamená „zahrnúť všetko“.

Štruktúra siete LSTM. Architektúry LSTM sú schopné naučiť sa dlhodobé závislosti v sekvenčných údajoch, vďaka čomu sú vhodné pre úlohy, ako je preklad jazyka, rozpoznávanie reči a prognózovanie časových radov.

Architektúra LSTM (Obr. 11) zahŕňa pamäťovú bunku, ktorá je riadená tromi bránami: vstupnou bránou, bránou zabudnutia a výstupnou bránou. Tieto brány rozhodujú o tom, aké informácie sa majú pridať, odobrať z pamäťovej bunky a aké informácie z pamäťovej bunky vydajú.

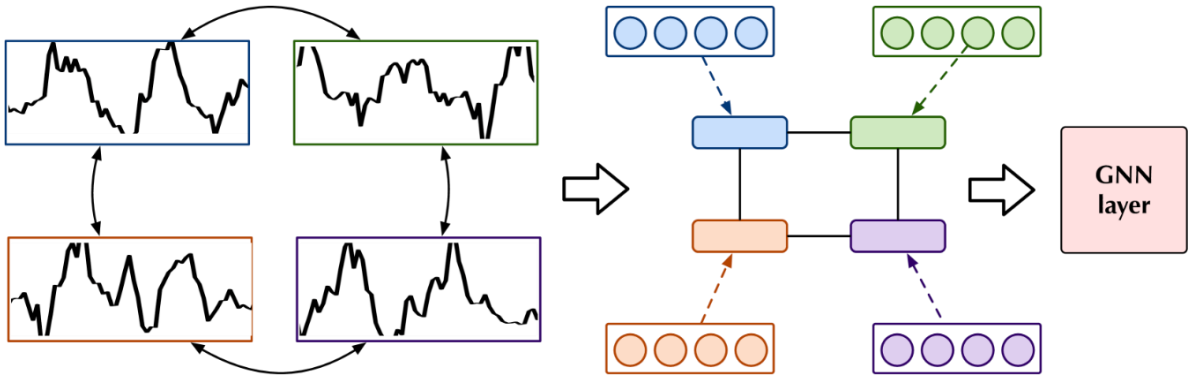
- Vstupná brána riadi, aké informácie sa pridávajú do pamäťovej bunky.
- Brána zabudnutia riadi, aké informácie sa odstránia z pamäťovej bunky.
- Výstupná brána riadi, aké informácie sú na výstupe z pamäťovej bunky.



Obrázok 11. Štruktúra siete LSTM.
Zdroj: (Tang, 2023)

Smerované grafové neurónové siete (Graph Neural Networks, GNN) sú určené na prácu s dátami reprezentovanými ako grafy, kde jednotlivé uzly predstavujú premenné a hrany vzťahy medzi nimi. Analýza údajov z viacerých premenných časových radov je často náročná z dôvodu zložitých a často nelineárnych korelácií medzi premennými. Na vyriešenie tohto problému boli v analýze časových radov široko používané GNN (Kipf & Welling, 2017).

Na Obr. 12 je znázornené využitie GNN pre modelovanie viacrozmerných časových radov so vzájomnými závislosťami. Viacrozmerné časové rady transformované na uzlové reprezentácie, ktoré následne spracováva vrstva GNN, čo umožňuje zachytiť komplexné vzťahy medzi jednotlivými dimenziami alebo entitami.



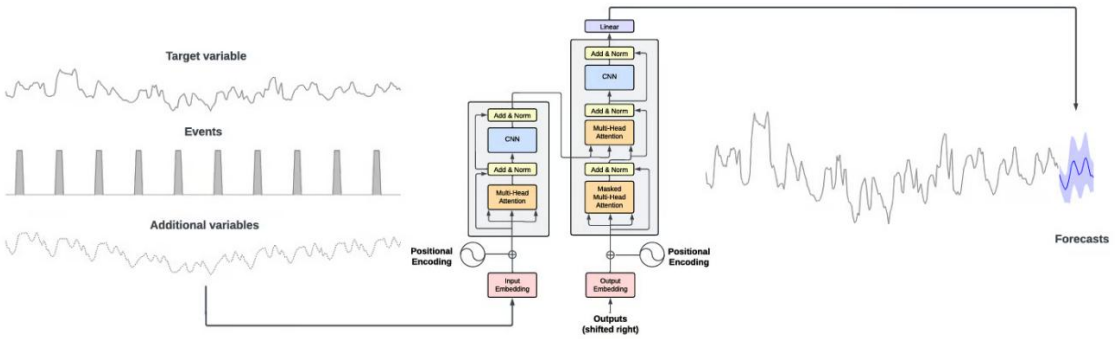
Obrázok 12: Ilustrácia modelovania viacrozmerných časových radov pomocou GNN.
Zdroj: Wang, 2024

1.3.4 Modely na báze transformerov GPT

Transformery predstavujú najnovší prístup k modelovaniu sekvenčných dát.

GPT (Generative Pretrained Transformer) je typ modelu hlbokého učenia, založený na transformerovej architektúre, ktorý je schopný generovať sekvencie výstupu na základe kontextu. Pôvodne bol navrhnutý pre úlohy spracovania prirodzeného jazyka, ale v súčasnosti sa objavujú aj jeho aplikácie v oblasti časových radov — napríklad TimeGPT.

TimeGPT je generatívny predtrénovaný model navrhnutý špeciálne na predikciu časových radov. Vychádza z transformerovej architektúry a bol natrénovaný na veľkom množstve verejných časových radov. Umožňuje predikovať budúce hodnoty cieľovej premennej s prihliadnutím na historické údaje, externé premenné a kalendárne udalosti (Awan, 2024).



Obrázok 13: Architektúra predikcie časových radov pomocou TimeGPT
Zdroj: Garza et al., 2024

Obrázok 13 znázorňuje základnú štruktúru modelu TimeGPT. Na vstupe sú cieľová premenná, kalendárne udalosti a doplnkové premenné, ktoré sú transformované do číselných reprezentácií pomocou vstupného zakódovania a pozíciového zakódovania. Vstupné zakódovanie (tzv. input embedding) zabezpečuje prevod kategórií alebo číselných hodnôt do viacrozmerného vektorového priestoru, zatiaľ čo pozíciové zakódovanie (positional encoding) zachytáva informáciu o poradí jednotlivých časových krokov v sekvencii. Následne prechádzajú sériou vrstiev typu CNN a multi-head attention. Na výstupe model generuje predikciu s odhadom neistoty.

Transformer architektúra (pôvodne vyvinutá na spracovanie prirodzeného jazyka) sa opiera o mechanizmus self-attention, ktorý vie modelovať vzťahy medzi ľubovoľnými dvoma pozíciami v sekvencii priamo. To znamená, že na rozdiel od RNN, ktoré idú postupne krok za krokom, transformer vidí časový rad skôr ako celok a učí sa, ktoré časové body sa navzájom ovplyvňujú.

Pre finančné dáta sa objavili varianty ako **Informer, Transformer-XL, Temporal Fusion Transformer**, atď. Štúdia z roku 2024 (Chen et al.) ukázala, že transformer-based modely (TSMixer, FEDformer, **Informer**) prekonali tradičné LSTM či TCN v predikcii kurzu RMB/USD.

1.3.5 Modely na základe fuzzy logiky

Fuzzy logika predstavuje alternatívny prístup k modelovaniu neurčitosti a aproximácii predikčných modelov, najmä v prípadoch, keď sú dáta nepresné, šumové alebo zahŕňajú kvalitatívne aspekty. Na rozdiel od tradičných pravdepodobnostných a štatistických metód fuzzy logika pracuje s pojmom neostrej množiny (**fuzzy set**), kde prvky nemajú binárne zaradenie („patrí“ alebo „nepatrí“), ale sú ohodnotené stupňom príslušnosti (**membership function**), ktorý nadobúda hodnoty v intervale ⟨0, 1⟩ (Zadeh, 1965).

V oblasti predikcie boli fuzzy množiny rozšírené na **fuzzy časové rady (Fuzzy Time Series – FTS)**, ktoré formálne definovali Song a Chissom (1993). Ich metóda vytvára fuzzy pravidlá na základe historických dát a umožňuje predikciu hodnôt bez potreby prísnych štatistických predpokladov, ako je stacionarita či normalita rozdelenia.

Tento prístup našiel uplatnenie napríklad v práci Degtiareva (2006), ktorý úspešne aplikoval fuzzy modely na predikciu výmenného kurzu USD/RUB.

Fuzzy inferenčné systémy (Fuzzy Inference Systems, FIS)

- Tieto modely využívajú pravidlá „ak-potom“ (**if-then rules**) na určenie vzťahov medzi premennými.

Existujú dva hlavné prístupy k fuzzy inferencii:

Mamdaniho model (logický) – Pracuje s fuzzy množinami nielen vo vstupoch, ale aj vo výstupoch. Výsledok inferencie sa získava pomocou *agregácie výstupných fuzzy množín* a ich následnej **defuzzifikácie**. Je vhodný pre systémy, kde je dôležité interpretovateľné rozhodovanie (napr. expertné systémy).

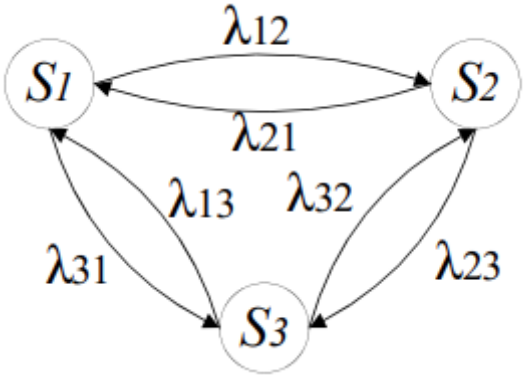
Sugeno model (funkčný) – Používa funkčné závery pravidiel, t. j. výstup každého pravidla je daný algebraickou funkciou vstupov (napr. lineárnou kombináciou). Tento model je presnejší pri výpočtoch a vhodný pre systémy automatického riadenia(Hudec, 2016).

Hybridné fuzzy modely

- **Fuzzy ARIMA/SARIMA**: Kombinácia fuzzy logiky a autoregresných modelov, kde fuzzy logika rieši neistotu v dátach a ARIMA modeluje štrukturálne vzory.
- **Fuzzy neurónové siete (FNN – Fuzzy Neural Networks)**: Spája fuzzy logiku s neurónovými sieťami na zvýšenie robustnosti modelu voči neistote (Hwang et al., 1998).
- **Fuzzy Bayesian Networks**: Umožňujú modelovanie pravdepodobnostných vzťahov v kontexte fuzzy pravidiel.

1.3.6 Modely založené na Markovom reťazci

Markovové reťazce (Markov Chain, MC) predstavujú matematický model, ktorý sa využíva aj pri predikcii systémov s náhodným správaním. Vďaka svojej schopnosti zachytiť dynamiku prechodov medzi stavmi systému sú Markovove reťazce vhodné na predpovedanie trendov na základe pravdepodobností. V oblasti predikcie sa často aplikujú v prípadoch, kde historické dáta umožňujú odhad prechodových pravdepodobností, ako napríklad v ekonómii na modelovanie zmien na trhu, alebo v biológii na predikciu vývoja procesov.



Obrázok 14: Markov reťazec s 3 stavmi
Zdroj: Ross, S. M., 1995

Vďaka schopnosti zachytiť dynamiku prechodov medzi stavmi sú MC vhodné na predpovedanie trendov v podmienkach náhodnosti a neistoty. Aplikujú sa napríklad v ekonómii, kde slúžia na modelovanie cyklických zmien na trhu (Hamilton, 1994) v biologických systémoch na modelovanie sekvencií (Durbin et al., 1998), ale aj v oblastiach ako predikcia správania zákazníkov, spracovanie prirodzeného jazyka a vývoj systémov umelej inteligencie. V predikcii časových radov sa môžu využívať aj skryté Markovove modely (Hidden Markov Models, HMMs), ktoré umožňujú pracovať s latentnými (nepozorovateľnými) stavmi systému a sú známe svojou robustnosťou v sekvenčných úlohách.

1.3.7 Model Prophet

Model Prophet nie je považovaný za klasický štatistický model, hoci využíva štatistické princípy. Ide o flexibilný nástroj na predikciu časových radov, ktorý bol vyvinutý spoločnosťou Facebook ako open-source riešenie. Tento model je obzvlášť vhodný na dáta s výraznými trendmi a sezónnosťou a ponúka intuitívne ovládanie, čo umožňuje rýchle ladenie a testovanie.

Hlavné charakteristiky modelu Prophet

1. Aditívny model: Prophet rozkladá časový rad na tri základné zložky:
    - Trend: Dlhodobé zmeny v dátach (lineárne alebo logistické).
    - Sezónnosť: Opakujúce sa vzory (denné, týždenné, ročné).
    - Sviatky: Špeciálne dni, ktoré majú vplyv na údaje.
2. Robustnosť: Model zvláda chýbajúce hodnoty, nepravidelné časové intervaly, výkyvy v údajoch aj náhle zmeny trendu. Vďaka tomu je vhodný na predikciu reálnych dát, ktoré zriedka spĺňajú ideálne podmienky klasických modelov.
3. Jednoduchosť použitia: Model vyžaduje minimálne predspracovanie dát a umožňuje rýchle pridávanie sezónnych vzorov alebo vlastných parametrov.

V porovnaní s modelmi ako ARIMA alebo exponenciálne vyrovnávanie poskytuje vyššiu flexibilitu a intuitívnejšie ovládanie, najmä pri výraznej sezónnosti a zložitých trendoch.

1.3.8 Porovnanie základných modelov predikcie

Rôzne modely predikcie časových radov ponúkajú odlišné výhody a obmedzenia, pričom ich výber závisí od charakteristík časového radu a požiadaviek na presnosť, interpretovateľnosť a zložitosť implementácie. Nasledujúca tabuľka porovnáva hlavné modely analyzované v tejto práci.

Tabuľka 2: Porovnanie modelov predikcie
Zdroj: vlastné spracovanie

Model	Opis	Obmedzenia
Lineárna regresia	Jednoduchá, rýchla, interpretovateľná	Citlivá na outliery, nevhodná pre nelineárne vzťahy
ARIMA/SARIMA	Vysoká presnosť pri stacionárnych dátach, dobre zvláda sezónnosť	Obmedzená presnosť pri vysokom stupni volatility a nepredvídateľných výkyvoch
ARCH/GARCH	Zachytáva volatilitu trhu, vhodné na analýzu rizík	Predpokladajú symetrický účinok šokov Lineárna štruktúra Neodhalenie štrukturálnych zmien
VAR	Vhodný – ak sa predikujú viaceré menové kurzy súčasne	Vyžaduje veľké množstvo historických dát
Simple RNN	Základná rekurentná architektúra	Slabšie učenie dlhodobých vzťahov (miznúci gradient)
LSTM	Zachytáva komplexné nelineárne vzťahy, vysoká presnosť	Veľká výpočtová náročnosť, potreba rozsiahlych datasetov
XGBoost	Sekvenčne trénované rozhod. stromy, dokáže odhaliť slabé vzory v dátach;	Riziko pretrénovania na staré trendy; veľa hyperparametrov; potreba veľa dát pre stabilnú generalizáciu.
ES	Vyhladzovanie časového radu s expo. váhami; Rýchle a nenáročné na výpočty	Nevhodné pre volatilné údaje a komplexné štruktúry
Prophet	Zvláda chýbajúce dáta, jednoduchá aplikácia	Menej presný pri vysoko volatilných dátach
Fuzzy logika	Pracuje s neistotou; Flexibilita pri modelovaní vzťahov	Subjektívnosť pri tvorbe pravidiel, slabá predikčná presnosť
MC	Modeluje pravdepodobné prechody medzi stavmi	Nevhodné pre kontinuálne alebo komplexné rady
GPT	Sieť založená na attention mechanizme, spracúva celú sekvenciu naraz . Veľmi vysoká presnosť pri dostatku dát;	Výpočtovo náročný; potrebuje obrovské množstvo tréningových dát;

2 Cieľ práce

Cieľom práce je - vytvoriť informačný systém, ktorý bude zbierať údaje o aktuálnom kurze vybraných mien, ukladať ich do databázy a na ich základe zobrazovať predikciu vývoja kurzov v najbližšom období.

Čiastkové ciele:

- analyzovať metódy predikcie vybraného časového radu
- analyzovať metódy tvorby matematického modelu jednotlivých kurzov
- analyzovať a následne aplikovať rôzne možností automatického ukladania údajov z vybraných stránok o aktuálnom kurze meny
- vytvoriť štandardnú internetovú aplikáciu, ktorá využíva PostgreSQL databázu
- implementovať automatizované procesy na zber a spracovanie údajov v reálnom čase s využitím vhodných technológií
- implementovať rôzne modely predikcie časových radov vrátane štatistických prístupov a metód strojového učenia
- porovnať presnosť a výkonnosť jednotlivých predikčných modelov na základe metrík
- vytvoriť vizualizácií reálnych a predikovaných údajov vo forme interaktívnych grafov na webovej platforme;

3 Metodika práce a metódy skúmania

Pri realizácii tohto projektu bolo potrebné stanoviť konkrétne metódy práce a postupy, ktoré zabezpečia efektívne riešenie stanovených cieľov. V tejto kapitole sa detailne opisuje postup pri návrhu jednotlivých komponentov systému, ako aj existujúce možnosti, ktoré je možné využiť pri zbere a spracovaní údajov a predikcii časových radov.

Na základe tejto analýzy boli zvolené konkrétne technológie a modely, ktoré najlepšie vyhovovali cieľom projektu a charakteru spracovávaných údajov. V nasledujúcich podkapitolách sú tieto možnosti predstavené spolu s odôvodnením výberu finálneho riešenia.

3.1 Zber a spracovanie údajov

Zdroje dát:

Primárnym zdrojom údajov je Európska centrálna banka (European Central Bank, ECB). ECB je centrálnou bankou devätnástich krajín Európskej únie krajín, ktoré prijali euro. Referenčné sadzby sa zvyčajne aktualizujú okolo 16:00 SEČ každý pracovný deň okrem dní pracovného pokoja ECB. Banka poskytuje bežné a historické eurá zahraničné referenčné výmenné kurzy pre 32 mien. Výmenné sadzby je možné stiahnuť vo forme súborov CSV, XML alebo PDF.

API Európskej centrálnej banky poskytuje aktuálne a historické kurzy vybraných mien. Prostredníctvom pravidelných požiadaviek sa údaje získavajú a následne ukladajú do databázy

Forexové portály a finančné služby. Alternatívne zdroje údajov, ktoré sprístupňujú historické a aktuálne kurzy vo formátoch ako CSV alebo JSON. V niektorých prípadoch slúžia ako záložné riešenie v prípade nedostupnosti údajov z oficiálnych inštitúcií.

Otvorené API rozhrania tretích strán. Napríklad služby ako Frankfurter.app alebo Open Exchange Rates, ktoré ponúkajú verejné API s podporou viacerých mien a jednoduchým prístupom k historickým dátam.

Na základe dostupnosti vyššie uvedených zdrojov boli v rámci práce zohľadnené a v niektorých prípadoch implementované tieto prístupy:

- **Prístup cez API rozhrania**. Umožňuje automatizovaný prístup k dátam vo formátoch ako XML alebo JSON, čo výrazne zjednodušuje ich ďalšie spracovanie.

- **Web scraping**. V prípadoch, kde nie je dostupné API, je možné údaje získavať automaticky z webových stránok prostredníctvom techniky web scrapingu. Tento prístup je však citlivejší na zmeny v štruktúre webu.
- **Import z tabuľkových súborov (napr. Excel)**. Niektoré zdroje sprístupňujú údaje vo forme Excel súborov, ktoré je možné ďalej spracovať pomocou skriptov alebo makier.
- **Automatizované skriptované riešenia**. Použitie vlastných skriptov na pravidelný zber, transformáciu a ukladanie údajov do databázy. Tento prístup umožňuje úplnú kontrolu nad kvalitou a frekvenciou aktualizácie údajov.

3.2 Návrh databázového systému

Metodika návrhu databázy zahŕňa rozdelenie celého procesu návrhu do niekoľkých fáz, z ktorých každá pozostáva z niekoľkých etáp. Všeobecne uznávaná metodika návrhu databázy je rozdelená do 3 hlavných fáz:

1. konceptuálny návrh
2. logický návrh
3. fyzický návrh

Konceptuálny model predstavuje abstraktný návrh databázovej štruktúry s dôrazom na identifikáciu entít, ich vlastností (atribúty) a vzťahov medzi nimi. Koncepčný návrh je nezávislý od akýchkoľvek podmienok fyzickej implementácie.

Logický návrh detailne špecifikuje štruktúru databázy pomocou racionálneho modelu. V tomto návrhu sú definované primárne a cudzie kľúče, dáta sú organizované v tabuľkách a sú normalizované na 3. normálnu formu.

Ide o proces konštrukcie informačného modelu založeného na existujúcich dátových modeloch bez ohľadu na použitý SRBD a ďalšie podmienky fyzickej implementácie.

Požiadavky na logický návrh:

- Tabuľky a vzťahy budú štruktúrované tak, aby sa zabezpečila konzistencia dát a minimalizovala redundancia.
- Normalizácia: Model bude normalizovaný do tretej normálnej formy (3NF), aby sa eliminovali zbytočné duplicity a zabezpečila integrita dát.
- Primárne a cudzie kľúče budú definované pre zabezpečenie integrity referenčných vzťahov medzi tabuľkami.

Fyzický model. Ide o postup vytvorenia popisu konkrétnej implementácie databázy s popisom štruktúry ukladania údajov a spôsobov prístupu k údajom.

Požiadavky na fyzický návrh:

- Tabuľky vytvorené v súlade s logickým modelom, pričom sa zohľadnia optimalizácie pre čítanie a zápis dát.
- Indexy: Definovanie primárnych a sekundárnych indexov pre urýchlenie prístupu k často používaným údajom, ako sú hodnoty kurzov a dátumy.
- Úložné zariadenia: Optimalizácia ukladania údajov na diskovom priestore s ohľadom na veľkosť databázy a požiadavky na výkon.
- Zálohovanie a replikácia: Implementácia mechanizmov pre zálohovanie a replikáciu dát, aby sa zabezpečila ich dostupnosť a ochrana pred stratou.

3.2.1 Výber SRBD

Systém riadenia bázy dát (SRBD) predstavuje základný komponent pri navrhovaní a implementácii informačného systému.

V kontexte práce boli zvažované predovšetkým relačné SRBD (RDBMS), keďže údaje o výmenných kurzoch majú dobre štruktúrovanú povahu a vzťahy medzi entitami (napr. mena, kurz, dátum) sú jasne definovateľné. Uvažované boli nasledujúce systémy:

- **MySQL** – otvorený, populárny SRBD s dobrou podporou pre webové aplikácie, využívaný napr. v kombinácii s PHP a Python.
- **PostgreSQL** – pokročilejší open-source SRBD s dôrazom na súlad so štandardom SQL a podporou komplexnejších dátových typov a operácií.
- **SQLite** – veľmi ľahký, vstavaný SRBD, vhodný pre prototypovanie alebo aplikácie s nízkymi nárokmi na súbežnosť a objem dát.
- **MariaDB** – fork MySQL s otvorenejším vývojovým modelom a podobnou syntaxou.

Ich porovnanie z hľadiska základných charakteristík je uvedené v nasledujúcej tabuľke:

Tabuľka 3: Porovnanie SRBD
Zdroj: vlastné spracovanie

Vlastnosť / Systém	MySQL	PostgreSQL	SQLite	MariaDB
Typ licencie	GPL	PostgreSQL	Public Domain	GPL/LGPL
Podpora SQL štandardu	Stredná	Vysoká	Základná	Stredná
Výkonnosť	Vysoká	Vysoká	Nízka	Vysoká
Podpora ORM (Django)	Áno	Áno (preferovaná)	Áno	Áno
Vhodnosť pre cloud	Dobrá	Výborná	Obmedzená	Dobrá

Na základe uvedeného porovnania však nie je možné jednoznačne určiť „najlepší“ SRBD bez ohľadu na zvyšok architektúry systému. Voľba konkrétneho databázového systému bude úzko súvisieť s výberom webového frameworku a zvoleným serverom alebo cloudovou platformou, na ktorej bude systém nasadený.

3.3 Modelovanie a predikcia výmenných kurzov: výber algoritmov

Výber vhodného matematického modelu je kľúčový pre presnosť predikcie a následné využitie týchto informácií pri rozhodovaní. Predikcia menových kurzov patrí medzi náročné úlohy v oblasti finančnej analýzy a modelovania, pretože výmenné kurzy sú ovplyvnené množstvom komplexných faktorov, medzi ktoré patrí ekonomická situácia, geopolitické udalosti, trhová volatilita a ďalšie makroekonomické premenné. V tejto kapitole sa zameriavame na popis hlavných modelov využívaných na predikciu menových kurzov.

Výber vhodného modelu na predikciu menových kurzov závisí od charakteru časového radu. V prípade stacionárnych časových radov s nízkou volatilitou je vhodné zvážiť použitie ARIMA alebo SARIMA modelov. Pre časové rady s vysokou volatilitou, typické pre menové trhy, sa ako najvhodnejšie ukazujú GARCH modely. Neurónové siete, najmä LSTM, sú preferovanou voľbou pre časové rady s komplexnou štruktúrou a nelineárnymi vzťahmi. V prípadoch, kde časový rad vykazuje sezónne a volatilné komponenty súčasne, môžu byť vhodné hybridné modely, ktoré kombinujú výhody viacerých prístupov.

Na základe teoretických východísk uvedených v kapitole 1.3 sme pristúpili k výberu modelov, ktoré sú najvhodnejšie na predikciu menových kurzov.

3.3.1 Analýza charakteristík časových radov

Pred aplikáciou konkrétneho predikčného modelu je nevyhnutné pochopiť základné vlastnosti časového radu, ktoré môžu výrazne ovplyvniť jeho správanie a predikčnú schopnosť použitých algoritmov.

Stacionarita

1. Konštantný priemer:

Rovnica 1:Konštantný priemer

$$E[X_t] = \mu, \quad \forall t \tag{1}$$

kde μ je konštantná stredná hodnota.

2. Konštantný rozptyl:

Rovnica 2: Konštantný rozptyl

$$Var(X_t) = \sigma^2, \ \ \forall t \tag{2}$$

kde σ^2 je konštantná variancia.

3. Autokovariancia závisí len od časového posunu *h* a nie od konkrétneho časového bodu *t*:

Rovnica 3: Autokovariancia

$$\mathrm{Cov}(\mathrm{X_t}, \mathrm{X_{t+h}}) = \gamma(\mathrm{h}), \ \ \forall \mathrm{t} \tag{3}$$

kde γ(h) je autokovariancia závislá len od vzdialenosti medzi dvoma hodnotami v časovom rade.

Testovanie stacionarity

Na overenie, či je časový rad stacionárny, sa používajú štatistické testy:

1. **Rozšírený test Dickeyho Fullera (Augmented Dickey-Fuller — ADF)**

ADF test — je rozšírením klasického Dickey-Fuller testu, ktorý slúži na overenie, či je časový rad stacionárny, teda či neobsahuje jednotkový koreň (*unit root*). ADF predpokladá model časoveho radu typu AR(p) a po odpočítaní y_{t-1} z oboch strán je reprezentovaný matematicky ako:

Rovnica 4: Rozšírený test Dickeyho Fullera

$$\Delta\, y_t = \delta y_{t-1} + \sum_{i=1}^{p} \beta_i \Delta y_{t-1} + \varepsilon_t \tag{4}$$

kde:

- δ: kľúčový parameter, ktorý testujeme,
- $\beta_i \Delta y_{t-1}$: oneskorené rozdiely na odstránenie autokorelácie,
- p: počet lagov (určuje sa na základe kritérií ako AIC, BIC, atď.).

Vzorec testovacej štatistiky je:

Rovnica 5: Vzorec testovacej štatistiky

$$t_{\hat{\beta}_i} = \frac{\hat{\beta}_i}{SE(\hat{\beta}_i)} \tag{5}$$

Test sa vykonáva za nasledujúcich predpokladov:

- Nulová hypotéza $H_0 : \delta = 0$: Rad je nestacionárny
- Alternatívna hypotéza $H_1 : \delta < 0$: Rad je stacionárny
- p-hodnota >0,05 Neodmietnutie (H0)
- p-hodnota <= 0,05 Prijať (H1)

Autokorelačnú funkciu (ACF) medzi veličinami y_t a y_{t-k} možno v prípade stacionárneho stochastického procesu y_t vyjadriť ako:

Rovnica 6: Autokorelačná funkcia

$$\rho_k = \frac{cov(y_t, y_{t-k})}{\sqrt{D(y_t)}\sqrt{D(y_{t-k})}} = \frac{\gamma_k}{\gamma_0} \quad (6)$$

kde:

- γ_k je autokovariačná funkcia definovaná takto:

Rovnica 7: Autokovariačná funkcia

$$\gamma_k = cov(y_t, y_{t-k}) = E(y_t - \mu_t)(y_{t-k} - \mu_t) \quad (7)$$

Ak autokorelačná funkcia (ACF) klesá pomaly, časový rad pravdepodobne nie je stacionárny. Ak ACF rýchlo klesá na nulu, časový rad je pravdepodobne stacionárny.

Transformácie na dosiahnutie stacionarity

Ak časový rad nie je stacionárny, môže byť transformovaný na stacionárny napríklad:

- Diferencovaním (First-order differencing)
- Logaritmickou transformáciou (na odstránenie exponenciálnych trendov).
- Odstránením sezónnych a trendových komponentov.

3.3.2 Výber vhodného predikčného modelu

Vzhľadom na komplexnosť časových radov menových kurzov neexistuje univerzálne optimálne riešenie. Preto boli testované viaceré modely a následne sme porovnávali ich výkonnosť, aby sme identifikovali najlepší model predikcie. Cieľom je vybrať taký model, ktorý bude najlepšie reflektovať dynamiku časových radov a poskytne presné a spoľahlivé predikcie. Z tohto dôvodu sa najskôr implementujú a testujú rôzne modely, ako sú ARIMA, GARCH, neurónové siete (RNN, LSTM) a modely ES.

1. Lineárna regresia

Rovnica 8: Model lineárnej regresie

$$y_t = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (8)$$

kde:

- *β0* je konštanta (intercept),
- *β1,β2,...,βk* sú regresné koeficienty
- ε(i): Náhodná zložka.

2. Prophnet

Ide o aditívny model pozostávajúci zo štyroch komponentov:

Rovnica 9: Model Prophnet

$$y_t = g(t) + s(t) + h(t) + \varepsilon_t\,, \tag{9}$$

kde:

- g(t): Trend časového radu.
- s(t): Sezónnosť (vplyv ročných období).
- h(t): Vplyv sviatkov.
- ε(t): Náhodná zložka.

3. ARIMA

Predtým, ako prejdeme k modelu ARIMA, zoznámime sa najskôr s ďalšími dvoma modelmi: kĺzavým priemerom a autoregresným modelom.

Model kĺzavého priemeru (MA): Model kĺzavého priemeru poriadku q opisuje závislosť medzi aktuálnou hodnotou časového radu a predchádzajúcimi chybami predikcie:

Rovnica 10: Model kĺzavého priemeru MA(q)

$$MA(q): y_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i\, \varepsilon_{t-i} \tag{10}$$

kde:

- y_t : hodnota časového radu v danom časovom okamihu t,
- μ : stredná hodnota časového radu (ak neexistuje trend),
- ε_t : náhodná chyba (biely šum) v čase t,
- θ_{i} : koeficienty modelu kĺzavého priemeru,
- q : poradie modelu, t.j. počet oneskorení náhodných chýb.

Potom pre model prvého rádu MA(q), kde q=1:

Rovnica 11: Model kĺzavého priemeru MA(1)

$$MA(1): y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} \tag{11}$$

Autoregresný model AR(p):

Rovnica 12: Autoregresný model AR(p)

$$AR(p):\ y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t \tag{12}$$

Potom pre model prvého rádu:

Rovnica 13: Autoregresný model AR(1)

$$AR(1):\ y_t = c + \varphi_1 y_{t-1} + \varepsilon_t \tag{13}$$

Integrovanie (I): zabezpečuje stacionaritu časového radu, čo je základnou podmienkou pre použitie ARIMA modelu. Diferencovanie sa používa na odstránenie nestacionarity:

Rovnica 14: Integrovanie

$$\Delta^d y_t = (1 - L)^d y_t \tag{14}$$

kde:

d je počet diferenciácií potrebných na dosiahnutie stacionarity a *L* je operátor posunu (lag). Kombináciou týchto dvoch modelov (autoregresie a kĺzavého priemeru) a transformačného kroku integrovania vzniká model **ARIMA(p, d, q):**

Rovnica 15: Model ARIMA(p, d, q)

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_{t-j} + \varepsilon_t \tag{15}$$

kde:

- y_t : hodnota časového radu v danom časovom okamihu t
- c : konštanta,
- ε_t : náhodná chyba (biely šum) v čase t,
- θ_{i} : koeficienty modelu kĺzavého priemeru,
- φ_i : koeficienty autoregresného komponentu.

Autoregresný model prvého rádu **(1,0,0)** - identický zápis ako AR(1).

Náhodná prechádzka (Random Walk):

Rovnica 16: Model ARIMA (0,1,0)

$$ARIMA(0,1,0):\ y_t = y_{t-i} + \varepsilon_t \tag{16}$$

Diferencovaný autoregresný model prvého radu:

Rovnica 17: Model ARIMA(1,1,0)

$$ARIMA(1,1,0):\ y_t = c + \varphi(y_{t-1} - y_{t-2}) + \varepsilon_t \tag{17}$$

Jednoduché exponenciálne vyhladzovanie:

Rovnica 18: Model ARIMA(0,1,1)

$$ARIMA(0,1,1): y_t = y_{t-i} + \theta\varepsilon_{t-1} + \varepsilon_t \quad (18)$$

Tento model je vhodný na modelovanie časového radu, kde sa predpokladá, že zmeny hodnôt závisia od náhodných chýb z predchádzajúcich období.

4. GARCH

Podľa Bollerslev (1986) **GARCH(p, q)**:

Rovnica 19:Model GARCH(p, q)

$$\sigma_t^2 = \omega \sum_{i=1}^{q} \alpha_i \, \varepsilon_{t-1}^2 + \sum_{j=1}^{p} \beta_i \, \sigma_{t-j}^2 \quad (19)$$

kde:

σ_t^2*:* podmienená variancia v čase t,

ω : konštantný člen (kladný),

αi : koeficienty vplyvu minulých chýb (ARCH efekt),

ε_{t-1}^2 : štvorcové hodnoty reziduálov v čase *t-i*,

βj : koeficienty vplyvu minulých variancií (GARCH efekt),

p : počet oneskorených variancií,

q : počet oneskorených kvadrátov chýb.

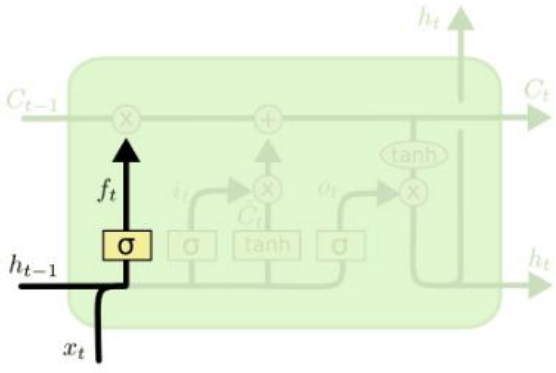
5. LSTM

Výpočty v rámci LSTM siete možno vyjadriť pomocou sústavy rovníc, ktoré popisujú činnosť jednotlivých brán a aktualizáciu vnútorného stavu bunky:

Zabúdajúca brána **(**Forget gate):

Rovnica 20: LSTM model - zabúdajúca brána

$$f_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_f\right) \quad (20)$$



Obrázok 15: Architektúra LSTM - forger gate
Zdroj: (Olah, 2015)

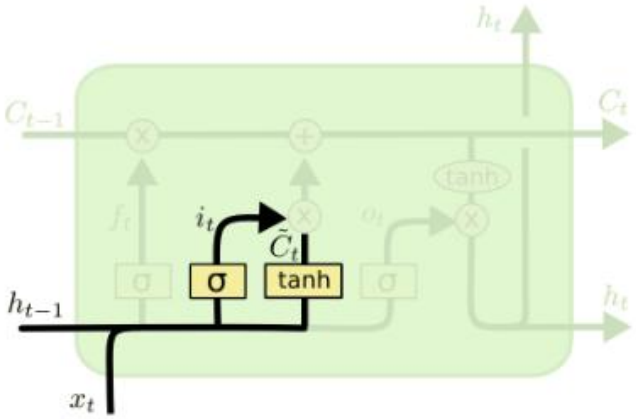
Vstupná brána:

Rovnica 21: LSTM model - Vstupný signál brány (input gate activation)

$$i_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_i\right) \tag{21}$$

Rovnica 22: LSTM model - Kandidátna hodnota bunky (cell input candidate)

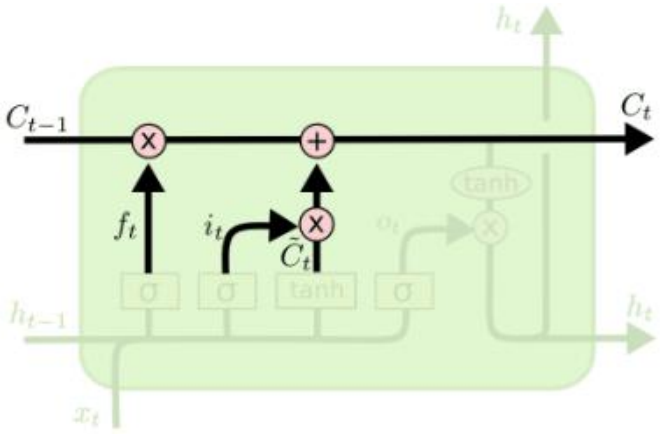
$$\tilde{C}_t = tahn(W_C * [h_{t-1}, x_t] + b_C) \tag{22}$$



Obrázok 16: Architektúra LSTM - input gate
Zdroj: (Olah, 2015)

Rovnica 23: LSTM model - Aktualizácia bunkového stavu

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{23}$$



Obrázok 17: Architektúra LSTM - aktualizácia bunkového stávu
Zdroj: (Olah, 2015)

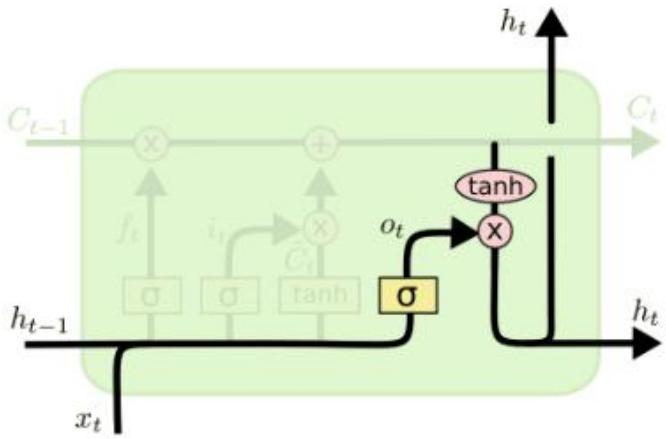
Výstupná brána (output gate):

Rovnica 24: LSTM model - Výstupná brána

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{24}$$

Rovnica 25: LSTM model - Výstup skrytej vrstvy

$$h_t = o_t * \tanh(C_t) \tag{25}$$



Obrázok 18: Architektúra LSTM - output gate
Zdroj: (Olah, 2015)

kde:

- x_t je vstup na čase t,
- ht−1 je výstup predchádzajúceho skrytého stavu,
- Ct je stav bunky na čase t,
- σ je sigmoidná aktivačná funkcia,
- W a b sú váhové matice a bias pre jednotlivé brány (Olah, 2015).

6. VAR

Rovnica 26: VAR model

$$y_t = A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_n y_{t-n} \varepsilon_t \tag{26}$$

kde:

- *Yt* je vektor k endogénnych premenných v čase t,
- A1,A2,...,An, sú matice koeficientov.

3.3.3 Miery presnosti prognóz

Pre hodnotenie presnosti predikčných modelov boli zvolené tri základné metriky: MAE, MSE a RMSE.

1. Priemerná absolútna chyba (Mean Absolute Error – **MAE**) — udáva priemernú absolútnu hodnotu chýb medzi skutočnými a predikovanými hodnotami. Jeho výhodou je jednoduchá interpretácia v pôvodných jednotkách merania a odolnosť voči odchýlkam.

Rovnica 27: MAE

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{27}$$

2. Priemerná kvadratická chyba (Mean Squared Error – **MSE**) — meria priemernú hodnotu druhých mocnín rozdielov medzi skutočnými a predikovanými hodnotami. Zvýrazňuje väčšie chyby, čo je užitočné pri detekcii modelov s výraznými odchýlkami.

Rovnica 28: MSE

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{28}$$

Kvadratická stredná chyba (Root Mean Squared Error – RMSE):

Rovnica 29: RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{29}$$

3.3.4 Softvérové nástroje pre matematické modelovanie a predikciu

Pre implementáciu vyššie spomenutých matematických modelov sa používajú nasledovné softvérové nástroje a knižnice:

- **Pandas**: Na manipuláciu s tabuľkovými údajmi, ich načítania, čistenia a analýzy.
- **NumPy**: Na spracovanie numerických operácií a prácu s veľkými dátovými štruktúrami.
- **Statsmodels**: Na tvorbu štatistických modelov a testovanie vlastností časových radov.
- **Scikit-learn**: Na predspracovanie údajov (poskytuje nástroje na škálovanie údajov - StandardScaler, MinMaxScaler) a hodnotenie presnosti modelov pomocou metrik, ako sú MAE, RMSE a MSE.
- **TensorFlow/Keras**: Na implementáciu pokročilých modelov neurónových sietí.
- **Prophet**: knižnica vyvinutá spoločnosťou Meta (Facebook) určená na modelovanie časových radov s výraznou sezónnosťou a trendovými zmenami.
- **TimeGPT (Nixtla)** – dostupná ako cloudová služba prostredníctvom knižnice nixtlats, umožňuje automatizovanú predikciu časových radov bez potreby manuálnej konfigurácie modelu.

Použitie týchto nástrojov umožňuje vytvárať robustné predikčné modely, ktoré dokážu zohľadniť štruktúru a charakteristiky konkrétneho časového radu, a tým zvýšiť spoľahlivosť predikcií.

3.4 Výber vývojových nástrojov pre webovú aplikáciu

V tejto časti sú popísané kľúčové riešenia pre backend, frontend a vizualizáciu údajov. Na základe analýzy viacerých možností boli vybrané tie riešenia, ktoré najviac zodpovedali požiadavkám projektu z hľadiska funkčnosti, flexibility a integrácie s predikčnými modelmi.

3.4.1 Backendové riešenia

Pre vývoj serverovej časti aplikácie bol ako hlavný programovací jazyk zvolený Python, vzhľadom na jeho široké využitie v oblasti analýzy údajov, strojového učenia a vývoja webových aplikácií. Python ponúka rozsiahlu komunitu a množstvo dostupných knižníc, čo výrazne urýchľuje vývoj a integráciu jednotlivých komponentov systému.

V rámci výberu webového frameworku boli porovnané dve najčastejšie využívané možnosti – **Flask** a **Django**.

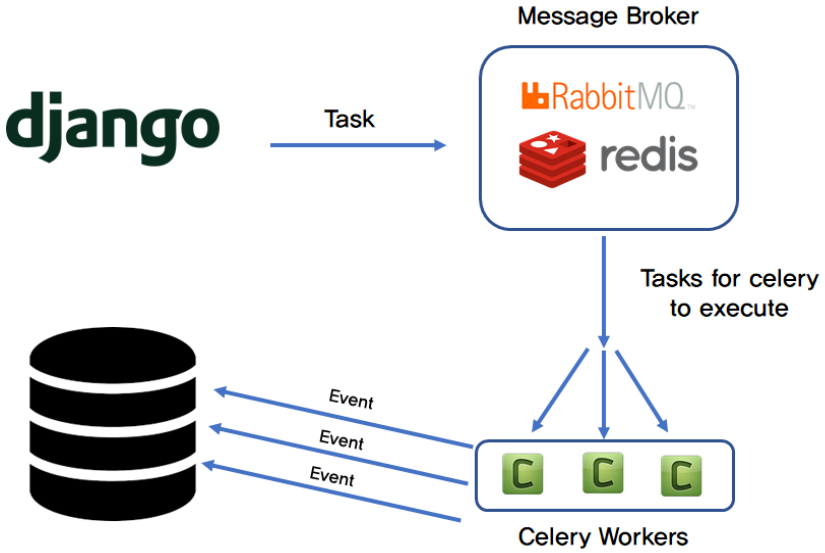
Flask predstavuje minimalistický a flexibilný mikroframework, ktorý poskytuje základnú štruktúru na vývoj webových aplikácií, pričom vývojár má veľkú mieru voľnosti pri výbere a integrácii doplnkových komponentov. Je vhodný najmä pre menšie alebo experimentálne projekty (Pallet, 2022).

Django oproti tomu predstavuje plnohodnotný rámec so „všetkým zabudovaným“ (angl. *batteries-included*), ktorý obsahuje vstavaný ORM, systém šablón, autentifikáciu, spracovanie formulárov, správu databáz a ďalšie komponenty. Vďaka týmto vlastnostiam výrazne urýchľuje vývoj robustných aplikácií a minimalizuje potrebu manuálnej konfigurácie. Django uľahčuje vývojárom implementovať komplexné funkcie s minimálnym kódom a poskytuje mnoho užitočných nástrojov, ktoré pokrývajú bežné potreby webových projektov (Django Documentation, 2023).

Celery je open-source nástroj na správu a vykonávanie asynchrónnych úloh v distribuovaných systémoch. Vo webovej aplikácii zabezpečuje Celery automatické zberanie údajov o kurzoch z API ECB v pravidelných intervaloch.

Redis je open-source databáza v pamäti, ktorá sa využíva predovšetkým na rýchle spracovanie a ukladanie údajov. Funguje ako medzipamäť (cache), ktorá zrýchľuje aplikácie tým, že eliminuje potrebu opakovaného načítania údajov z pomalších databáz.
Redis slúži ako správca správ (angl. **message broker)** na koordináciu úloh medzi Django aplikáciou a Celery. Jeho schopnosť rýchlo manipulovať s frontami úloh a spravovať veľké množstvo krátkodobých údajov robí z neho ideálny nástroj na spracovanie údajov v reálnom

čase. Výhodou Redis je jeho rýchlosť, jednoduchá implementácia a schopnosť zvládať vysoké zaťaženie, čo je kľúčové pri spracovávaní veľkých objemov dát. Princíp fungovania systému Celery v spojení s Redis je znázornený na Obr. 19.



Obrázok 19: Architektura Celery
Zdroj: Medium, 2023

3.4.2 Frontend technológie

Pri výbere technológií pre tvorbu používateľského rozhrania boli analyzované viaceré možnosti s ohľadom na jednoduchosť implementácie, responzívnosť dizajnu a kompatibilitu s backendovým riešením.

- **Hypertextový značkovací jazyk (HyperText Markup Language, HTML)** je značkovací jazyk používaný na tvorbu štruktúry webových stránok. Slúži ako základ pre definovanie obsahu stránky, vrátane textov, obrázkov a odkazov. Je to nevyhnutný komponent každej webovej aplikácie (W3Schools, 2023).
- **Kaskádové štýly (Cascading Style Sheets, CSS)** predstavujú štýlovací jazyk, ktorý dizajnérom poskytuje prostriedky na formátovanie a oživenie vzhľadu prvkov HTML. Vďaka CSS možno vytvárať vizuálne príťažlivé a plne prispôsobivé používateľské rozhrania (MDN Web Docs, 2023).
- **Bootstrap** je CSS framework, ktorý ponúka preddefinované komponenty, ako sú tlačidlá, formuláre a mriežky. Tento framework sme zvolili pre jeho robustnosť a schopnosť rýchlo vytvárať dizajny, optimalizované pre rôzne typy zariadení (Bootstrap Documentation, 2023).
- **JavaScript:** JavaScript je skriptovací jazyk na pridávanie interaktívnych prvkov na webové stránky. Používa sa na dynamickú manipuláciu s obsahom stránky, ako

napríklad aktualizáciu grafov alebo spracovanie užívateľských vstupov v reálnom čase (Eloquent JavaScript, 2023).

Tieto technológie spolu vytvárajú základ pre moderné, responzívne a interaktívne používateľské rozhranie, ktoré je plne integrované s funkčnosťou serverovej časti systému.

3.4.3 Vizualizačné nástroje

Pre vizualizáciu dát sa používajú knižnice **Plotly.js a Matplotlib**. Plotly.js je knižnica pre JavaScript, ktorá umožňuje vytvárať interaktívne grafy a vizualizácie dát. Táto knižnica je ideálna pre zobrazenie časových radov a predikcií, pričom ponúka funkcie ako zoomovanie, posúvanie a zobrazenie detailov (Plotly Documentation, 2023).

Matplotlib je jedna z najznámejších a najpoužívanejších knižníc pre vizualizáciu dát v Pythone. Matplotlib sme zvolili najmä pre **analytickú fázu projektu**. Na rozdiel od Plotly.js neposkytuje natívnu interaktivitu pre webové aplikácie, je ideálna pre tvorbu profesionálnych, statických vizualizácií v rámci analytickej časti projektu alebo v Jupyter notebookoch (Matplotlib Documentation, 2023).

3.4.4 Integrované vývojové prostredie

Na vývoj a implementáciu informačného systému bol zvolený integrovaný vývojový nástroj **PyCharm Professional**, ktorý je jedným z najpoužívanejších IDE (Integrated Development Environment) pre programovanie v jazyku Python. PyCharm bol vyvinutý spoločnosťou **JetBrains.**

PyCharm Professional bol zvolený ako primárne vývojové prostredie, nakoľko spoločnosť JetBrains poskytuje študentom a pedagogickým pracovníkom z akreditovaných vzdelávacích inštitúcií bezplatnú vzdelávaciu licenciu. Táto licencia je dostupná pre študentov zapísaných do akreditovaných študijných programov s dennou formou výučby a umožňuje plnohodnotné využitie všetkých funkcionalít profesionálnej verzie PyCharm.

Okrem PyCharm Professional sa na vývoj a testovanie predikčných modelov používa aj **Jupyter Notebook**. Jupyter Notebook poskytuje interaktívne prostredie vhodné na analýzu údajov, testovanie modelov a vizualizáciu výsledkov. V rámci práce sa Jupyter využíva najmä na exploratívnu analýzu dát (EDA), testovanie rôznych predikčných algoritmov a ladenie parametrov modelov pred ich implementáciou do backendu systému. Tento prístup umožňuje oddelenie vývoja predikčných modelov od implementácie aplikácie, čím sa optimalizuje vývojový proces a zabezpečuje vyššia flexibilita pri výbere vhodného modelu.

4 Výsledky práce

Táto kapitola sa zameriava na praktickú realizáciu webovej aplikácie na zber a spracovanie údajov o valutových kurzoch, vychádzajúc z teoretických poznatkov a metodických prístupov prezentovaných v predchádzajúcich častiach práce. Opisuje návrh a implementáciu jednotlivých komponentov systému, vrátane automatizovaného získavania dát, ich ukladania do databázy a predspracovania na účely ďalšej analýzy, vývoj predikčných modelov, ich vyhodnotenie, ako aj samotnú implementáciu webového rozhrania.

4.1. Konfigurácia technologických nástrojov pre webovú aplikáciu

Na začiatku realizácie bolo potrebné vykonať konfiguráciu technologických nástrojov, ktoré zabezpečia stabilitu a efektívnosť celej aplikácie.

Na vytvorenie webovej aplikácie bol použitý Django framework verzie 5.1.4. Pred inicializáciou projektu bolo potrebné nainštalovať framework prostredníctvom správcu balíkov pip. Inštalácia prebehla pomocou nasledujúceho príkazu:

```
pip install django
```

Projekt bol inicializovaný v PyCharm Professional pomocou príkazu:

```
django-admin startproject exchange_rates
```

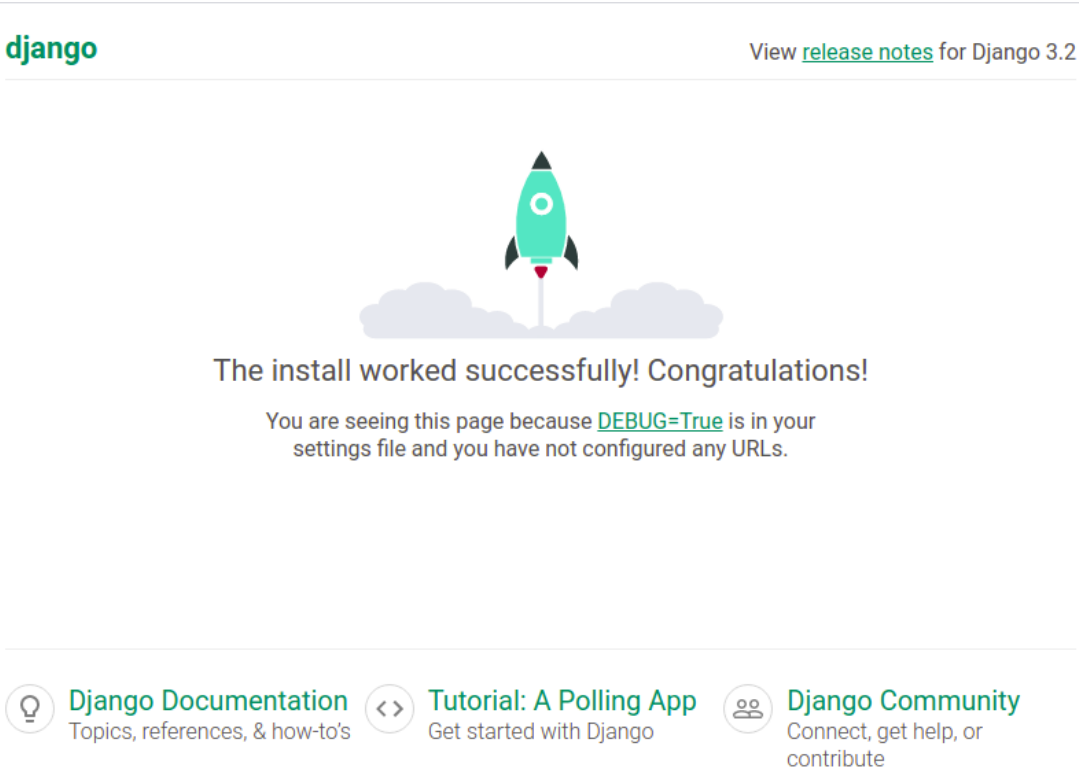
Súbor django-admin.exe poskytuje možnosť spúšťať množstvo príkazov na správu projektu Django. Na vytvorenie projektu sa používa najmä príkaz startproject. Tomuto príkazu sa ako argument odovzdá názov projektu. Webová aplikácia alebo projekt Django sa skladá z jednotlivých aplikácií. Spolu tvoria plnohodnotnú webovú aplikáciu. Každá aplikácia predstavuje špecifickú funkcionalitu alebo skupinu funkcií. Výsledkom bolo vytvorenie základnej štruktúry adresárov a súborov, ktorá zahŕňa najmä:

`manage.py` – hlavný skript na správu projektu, adresár s názvom projektu (`exchange_rates`), obsahujúci hlavné konfiguračné súbory, ako napríklad `settings.py`, `urls.py` a `wsgi.py`.

V počiatočnej fáze projekt neobsahoval žiadnu konkrétnu aplikáciu, napriek tomu bolo možné spustiť projekt. Vo vývojovom prostredí PyCharm s projektom „exchange_rates“ bol použitý príkaz:

```
python manage.py runserver
```

Ak sa lokálny server spustil bez chýb, bol dostupný na adrese http://127.0.0.1:8000/, kde sa zobrazí úvodná webová stránka Django.



Obrázok 20: Úvodná webová stránka Django
Zdroj: vlastné spracovanie

4.1.1 Vytvorenie aplikácie na Django

Pri vytvorení projektu Django automaticky obsahuje niekoľko vstavaných aplikácií, ktoré zabezpečujú základnú funkcionalitu frameworku. Ich zoznam je uvedený v nastaveniach projektu settings.py v sekcii INSTALLED_APPS.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

]
```

V rámci projektu vytvorená aplikácia „rates“ na zber a spracovanie dát:

```
python manage.py startapp rates
```

4.1.2 Nastavenie bázy dát

V Django sa databáza SQLite vytvára automaticky (štandardne), vytvorenie tejto konkrétnej databázy je špecifikované v konfiguračnom súbore projektu `settings.py`:

```
DATABASES={
    'default':{
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'os.path.join(BASE_DIR,'db.sqlite3')'
     }
}
```

Túto konfiguráciu je možné zmeniť na inú databázu, v našom prípade PostgreSQL, ktorú sme použili v súlade s cieľmi a metodikou práce. Pre použitie inej SRBD je potrebné nainštalovať príslušný balík, napríklad pre PostgreSQL je potrebné použiť nasledovný príkaz:

```
pip install psycopg2
```

Tabuľka: Prehľad základných SRBD kompatibilných s Django

Zdroj:

SRBD	Package	Príkaz inštalácie
PostgreSQL	psycopg2	pip install psycopg2
MySql	mysql-python	pip install mysql-python
Oracle	cx_Oracle	pip install cx_Oracle

Následne je potrebné upraviť konfiguráciu databázy v súbore settings.py nasledovne:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'currency_db',                  # Имя базы данных
        'USER': 'root',                         # Пользователь базы данных
        'PASSWORD': '',                         # Пароль
        'HOST': 'localhost',                    # Адрес сервера базы данных
        'PORT': ' 5432',
    }
}
```

Obrázok 21: Konfigurácia pripojenia k databáze PostgreSQL v súbore settings.py
Zdroj: vlastné spracovanie

4.1.3 Návrh databázovej štruktúry pre zber a ukladanie údajov

Objektovo-relačné mapovanie (Object-Relational Mapping, ORM) je technológia, ktorá prepája objektovo-orientovaný kód s relačnou databázou. V prípade Django zabezpečuje ORM prenositeľnosť modelov medzi rôznymi databázami.

Pri návrhu tabuľky ExchangeRate bola zvolená forma širokej tabuľky (wide table), kde jednotlivé meny sú reprezentované samostatnými stĺpcami. Tento prístup umožňuje jednoduché a rýchle načítanie viacerých mien v rámci jedného dotazu pri práci so surovými

dátami. Pre potreby predikčných modelov bola však vytvorená aj tabuľka ExchangeRateNormalized, kde sú kurzy normalizované a štruktúrované vo forme úzkych tabuliek (long table) s väzbou na tabuľku Currency.

Tabuľka 4: Implementácia databázovej schémy pomocou Django modelov
Zdroj: vlastné spracovanie

```python
class ExchangeRate(models.Model):
    id = models.AutoField(primary_key=True)  # ID, уже AUTO_INCREMENT
    date = models.DateField()  # Поле "date"
    USD = models.DecimalField(max_digits=10, decimal_places=4, null=True, blank=True)
    CNY = models.DecimalField(max_digits=10, decimal_places=4, null=True, blank=True)
    HUF = models.DecimalField(max_digits=10, decimal_places=4, null=True, blank=True)
    PLN = models.DecimalField(max_digits=10, decimal_places=4, null=True, blank=True)
    CZK = models.DecimalField(max_digits=10, decimal_places=4, null=True, blank=True)
    GBP = models.DecimalField(max_digits=10, decimal_places=4, null=True, blank=True)

    class Meta:
        db_table = 'exchange_rates'
```

```python
# Таблица для нормализованных курсов
class ExchangeRateNormalized(models.Model):
    date = models.DateField()
    # Указываем внешнюю связь на единственную модель Currency
    currency = models.ForeignKey(Currency, on_delete=models.CASCADE, db_column="currency_code"
    rate_value = models.DecimalField(max_digits=10, decimal_places=4)

    class Meta:
        db_table = 'exchange_rates_normalized'
        unique_together = ('date', 'currency')
```

```python
# Объединённое определение модели Currency
class Currency(models.Model):
    currency_code = models.CharField(max_length=3, primary_key=True)
    currency_name = models.CharField(max_length=50)

    class Meta:
        db_table = 'currencies'
```

```python
# Предсказания
class Prediction(models.Model):
    date = models.DateField()
    currency_code = models.ForeignKey(Currency, on_delete=models.CASCADE, db_column="currency_code"
    predicted_value = models.DecimalField(max_digits=10, decimal_places=4)
    model_name = models.CharField(max_length=50)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'predictions'
```

Aj keď Django ORM výrazne uľahčuje prácu s databázou a umožňuje manipuláciu bez explicitného písania SQL dotazov, pre úplnosť uvedieme príklady ekvivalentných SQL príkazov.

Vkladanie nového kurzu:

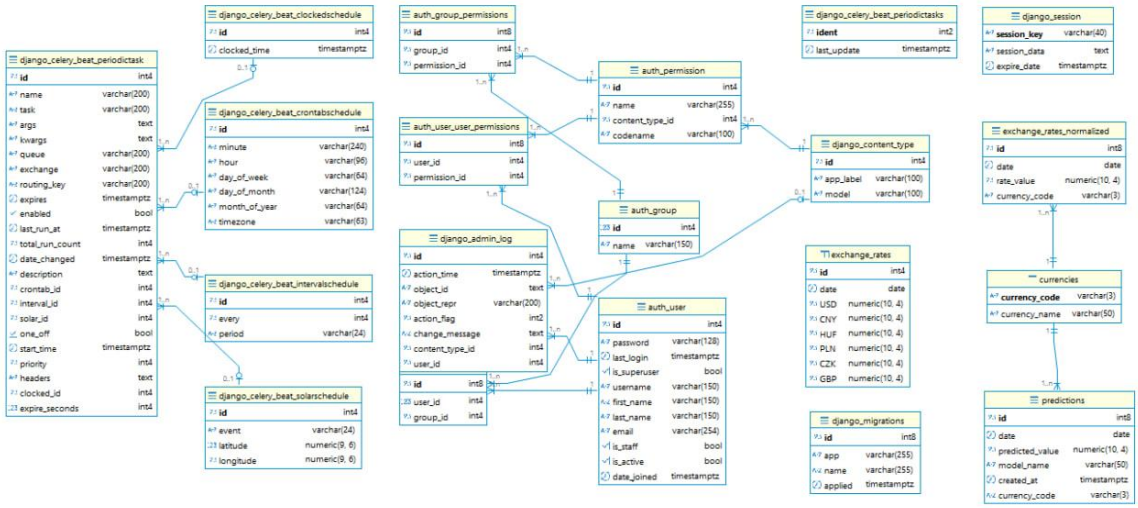
```
INSERT INTO rates_exchangerate (currency_id, date, rate)
VALUES (1, '2025-03-20', 1.0654);
```

Aktualizácia existujúceho kurzu:

```
UPDATE rates_exchangerate
SET rate = 1.0754
WHERE currency_id = 1 AND date = '2025-03-20';
```

Vytváranie tabuľky

```
CREATE TABLE exchange_rates (
    date DATE PRIMARY KEY,
    USD DECIMAL(10,4),
    GBP DECIMAL(10,4),
    CZK DECIMAL(10,4),
    HUF DECIMAL(10,4),
    PLN DECIMAL(10,4),
    CNY DECIMAL(10,4)
);
```



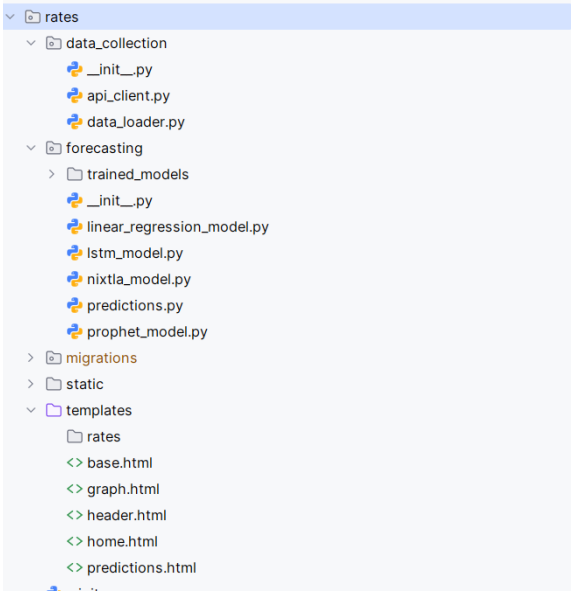
Obrázok 22: Fyzický model
Zdroj: vlastné spracovanie - DBeaver

Schéma fyzického modelu databázy obsahuje okrem aplikačných tabuliek (currencies, exchange_rates, exchange_rates_normalized, predictions) aj systémové tabuľky, ktoré boli automaticky vygenerované v rámci platformy Django a plánovača úloh Celery. Tieto systémové tabuľky (auth_user, django_admin_log, django_celery_beat_periodictask a ďalšie) zabezpečujú správu používateľských účtov, autentifikáciu, plánovanie periodických úloh a interné operácie frameworku.

4.1.4 Štruktúra webovej aplikácie

Pri vytvorení aplikácie pomocou frameworku Django dochádza k automatickému vygenerovaniu niekoľkých základných komponentov, ktoré tvoria základ aplikačnej štruktúry:

migrations/ – obsahuje automaticky generované súbory migrácií, ktoré umožňujú synchronizáciu modelov aplikácie s databázovou schémou. Tieto súbory sú vytvárané nástrojom Django ORM pri každej zmene modelov.



Obrázok 23: Štruktúra aplikácie "rates"
Zdroj: vlastné spracovanie - PyCharm

init.py – informuje interpret Pythonu, že aktuálny adresár bude považovať za balík;

admin.py – predpripravený súbor určený na registráciu modelov v administračnom rozhraní;

apps.py – definuje konfiguráciu aplikácie;

models.py – východiskový súbor pre definovanie dátových modelov;

tests.py – súbor pre písanie testovacích prípadov na overenie správnosti fungovania aplikácie, ktorý je rovnako súčasťou preddefinovanej štruktúry Django.

Pre zvýšenie prehľadnosti a udržateľnosti kódu bola aplikácia ďalej rozdelená do viacerých priečinkov, ktoré zodpovedajú jednotlivým funkčným častiam projektu:

data_collection/ – modul zodpovedný za zber údajov z externých API.

forecasting/ – modul pre predikciu výmenných kurzov pomocou rôznych modelov:

- linear_regression_model.py
- lstm_model.py – predikcia pomocou LSTM neurónovej siete.
- nixtla_model.py – predikcia pomocou TimeGPT modelu.
- prophet_model.py – predikcia pomocou modelu Prophet.
- predictions.py – agregácia a výstup predikčných výsledkov.

templates/rates/ HTML šablóny pre webové rozhranie aplikácie.

Týmto bola úspešne pripravená technologická infraštruktúra aplikácie, ktorá umožnila pristúpiť k ďalším krokom vrátane zberu údajov a implementácie modelov predikcie.

4. 2 Proces zberu dát

V rámci implementácie boli využité dve hlavné metódy získavania dát. Oba prístupy sa použili v rôznych fázach vývoja a umožnili flexibilnú prácu so získavaním historických a aktuálnych údajov.

4.2.1 Automatizovaný zber dát pomocou Django a Celery

Automatizovaný zber dát bol navrhnutý a implementovaný ako súčasť backendového riešenia v rámci webovej aplikácie. Na zabezpečenie plánovania a asynchrónneho vykonávania úloh na pozadí bol využitý nástroj Celery, ktorý umožňuje plánovanie a vykonávanie úloh na pozadí.

Princíp fungovania systému:

1. **Získavanie kurzových údajov** – údaje o výmenných kurzoch sú získavané priamo z API Európskej centrálnej banky (ECB), ktoré poskytuje dáta vo formáte XML. Na tento účel boli použité knižnica `requests` na sťahovanie obsahu a `xml.etree.ElementTree` na spracovanie XML odpovedí.
2. **Spracovanie a validácia dát** – Vzhľadom na skutočnosť, že ECB neposkytuje nové kurzové údaje počas víkendov a štátnych sviatkov, bolo nevyhnutné zahrnúť do logiky zberu dát mechanizmus na overovanie dostupnosti údajov. V rámci implementácie bol preto vytvorený zoznam oficiálnych sviatkov pre roky 2024 a 2025, ktorý v kombinácii s kontrolou kalendárnych dní (víkendov) umožňuje systému identifikovať posledný platný pracovný deň. Tým sa zabezpečuje, že aj v prípade absencie nových údajov z ECB API je možné zabezpečiť úplnosť dát.
3. **Automatizácia s Celery** – Pre správne fungovanie plánovania úloh bola vytvorená konfigurácia Celery v súbore celery.py, ktorý nastavuje prostredie a automaticky načítava úlohy (Obr. 24).

celery.py × tasks.py api_client.py settings.py

```
1  from __future__ import absolute_import, unicode_literals
2  import os
3  from celery import Celery
4
5  os.environ.setdefault( key: 'DJANGO_SETTINGS_MODULE', value: 'exchange_rates.settings')
6
7  app = Celery('exchange_rates')
8
9  app.config_from_object( obj: 'django.conf:settings', namespace='CELERY')
10 app.autodiscover_tasks()
11
```

Obrázok 24: Konfigurácia Celery v projekte
Zdroj: vlastné spracovanie

4. **Ukladanie údajov do databázy** – Získané údaje sú následne pripravené na uloženie do databázy PostgreSQL. Funkcia vždy vracia aktuálny dátum, aj v prípade, že v daný deň nie sú k dispozícii nové údaje z ECB, čím je zabezpečená konzistencia dátového záznamu v databáze.

Samotná úloha na zber kurzových údajov bola definovaná v tasks.py (Obr. 25), kde je popísaná logika získavania a ukladania dát.

celery.py | tasks.py | api_client.py | settings.py

```python
import ...

# Валюты, которые должны сохраняться
ALLOWED_CURRENCIES = {"USD", "CNY", "HUF", "PLN", "CZK", "GBP"}


@shared_task
def collect_and_save_exchange_rates():
    try:
        # Получение данных
        date, rates = fetch_exchange_rates()

        # Сохранение данных в ненормализованную таблицу
        save_exchange_rates(date, rates)

        # Сохранение данных в нормализованную таблицу
        for currency_code, rate_value in rates.items():
            if currency_code in ALLOWED_CURRENCIES:
                currency, _ = Currency.objects.get_or_create(currency_code=currency_code)
                ExchangeRateNormalized.objects.update_or_create(
                    date=date,
                    currency=currency,
                    defaults={'rate_value': rate_value}
                )

        return f"Данные за {date} успешно сохранены в обе таблицы!"
    except Exception as e:
        return f"Ошибка при выполнении задачи: {e}"
```

Obrázok 25: Definícia úlohy na zber údajov v tasks.py
Zdroj: vlastné spracovanie

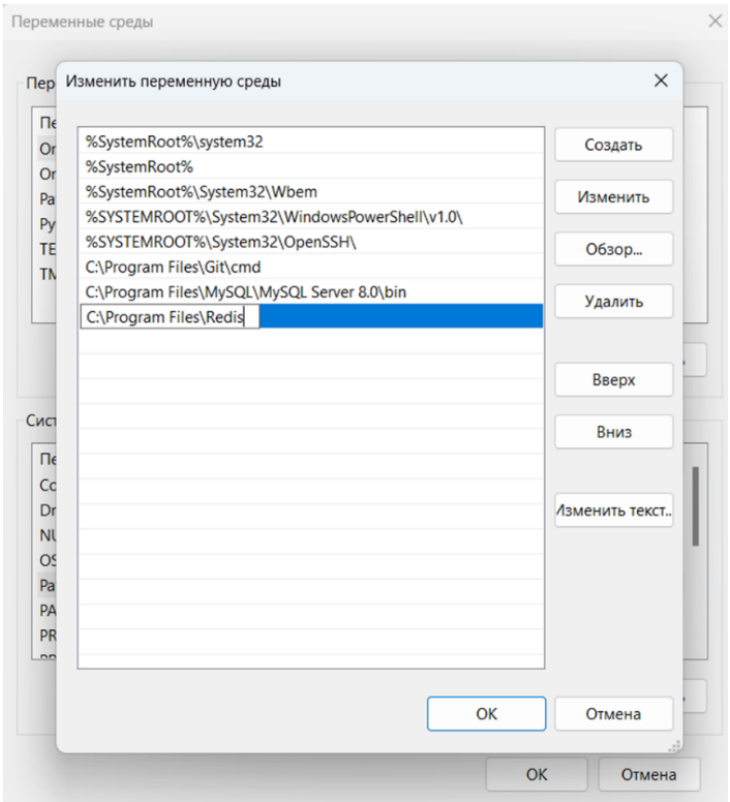
Funkcia `collect_and_save_exchange_rates` obsahuje viacvrstvovú logiku – najprv ukladá dáta v pôvodnom formáte a následne ich synchronizuje aj v normalizovanej podobe podľa meny. Kód navyše ošetruje výnimky a v prípade chyby v procese zaznamenáva informáciu o zlyhaní.

Inštalácia a konfigurácia Redis pre Celery

Redis bol použitý ako **message broker** na správu úloh v Celery. Na Obr. 22 je znázornený inštalačný proces Redis na Windows.

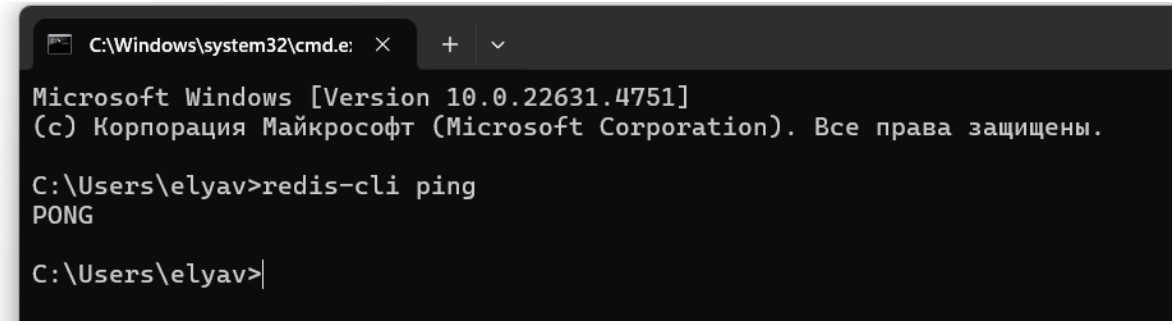
Obrázok 22: Inštalacia Redis na Windows
Zdroj: vlastné spracovanie

Po úspešnej inštalácii je potrebné pridať cestu k Redis do **systémových premenných**, aby bol správne rozpoznaný v systéme.



Obrázok 23: Konfigurácia systémových premenných
Zdroj: vlastné spracovanie

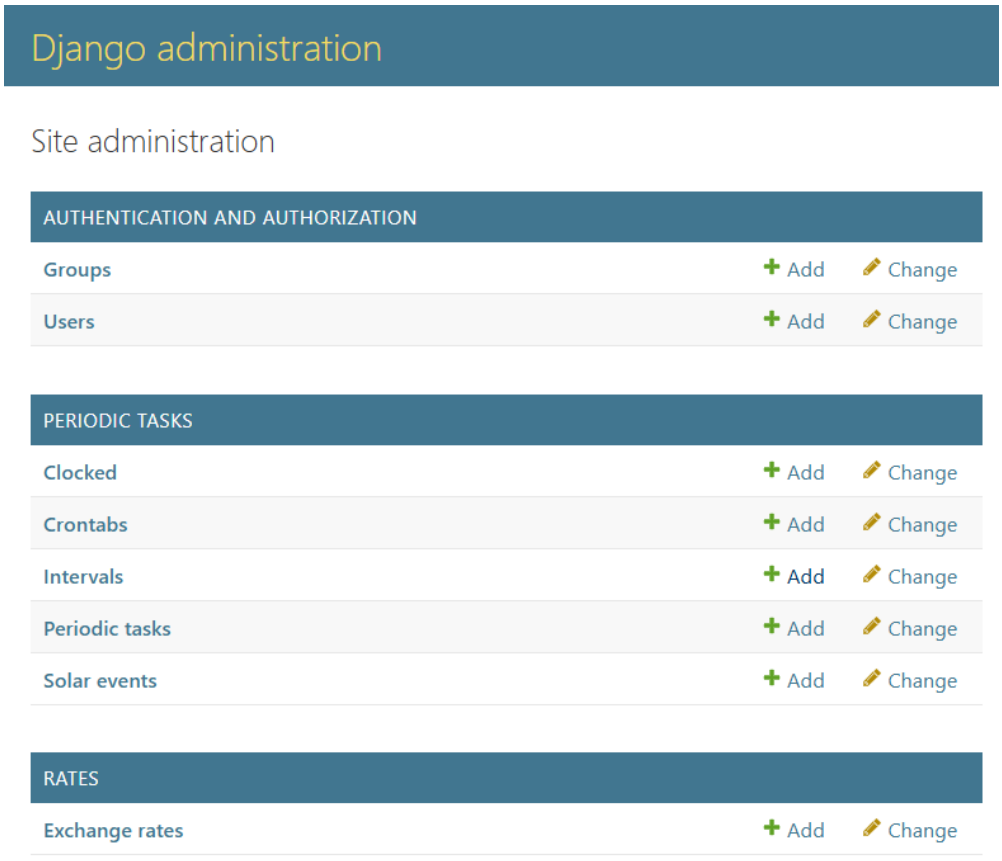
Výstup príkazu redis-cli ping, ktorý vracia „PONG“, čo znamená, že Redis beží správne.



Obrázok 26: Overenie Redis v CMD
Zdroj: vlastné spracovanie

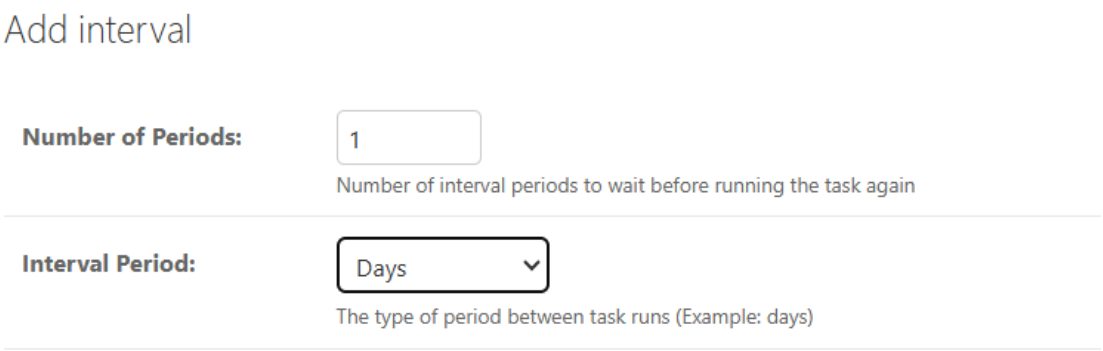
Administrácia Django aplikácie a nastavenie Celery Beat

Django administrácia umožňuje správu úloh a modelov priamo cez webové rozhranie. Nasledujúce snímky obrazovky ukazujú konfiguráciu plánovaných úloh a intervalov ich spúšťania. Na Obr. 27 je ukázané admin rozhranie Django, kde môžeme vidieť správu užívateľov, skupín a hlavne sekciu „Periodic Tasks“, ktorá obsahuje nastavenia pre plánované úlohy.

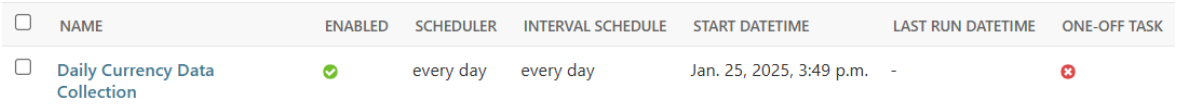


Obrázok 27: Django administrácia – hlavná stránka
Zdroj: vlastné spracovanie

Nastavenie intervalov v Django Celery Beat.



Obrázok 28: Formulár na pridanie intervalu pre plánovanú úlohu
Zdroj: vlastné spracovanie



Obrázok 29: Ukážka plánovanej úlohy „Daily Currency Data Collection"
Zdroj: vlastné spracovanie

Aby systém správne vykonával automatizované úlohy, je potrebné spustiť dva samostatné procesy:

- **Celery worker** — zodpovedá za vykonávanie naplánovaných úloh na pozadí. Worker spracúva úlohy, ktoré sú uložené v message brokeri (v tomto prípade Redis), a zabezpečuje ich vykonanie.
- **Celery Beat** — plánovač úloh, ktorý v definovaných intervaloch odosiela úlohy do fronty na spracovanie. V našej aplikácii bol použitý na zabezpečenie pravidelného zberu kurzových údajov.

Oba procesy musia byť spustené súčasne, aby systém automatizácie fungoval správne: Celery Beat plánuje úlohy a worker ich následne spracúva a vykonáva.

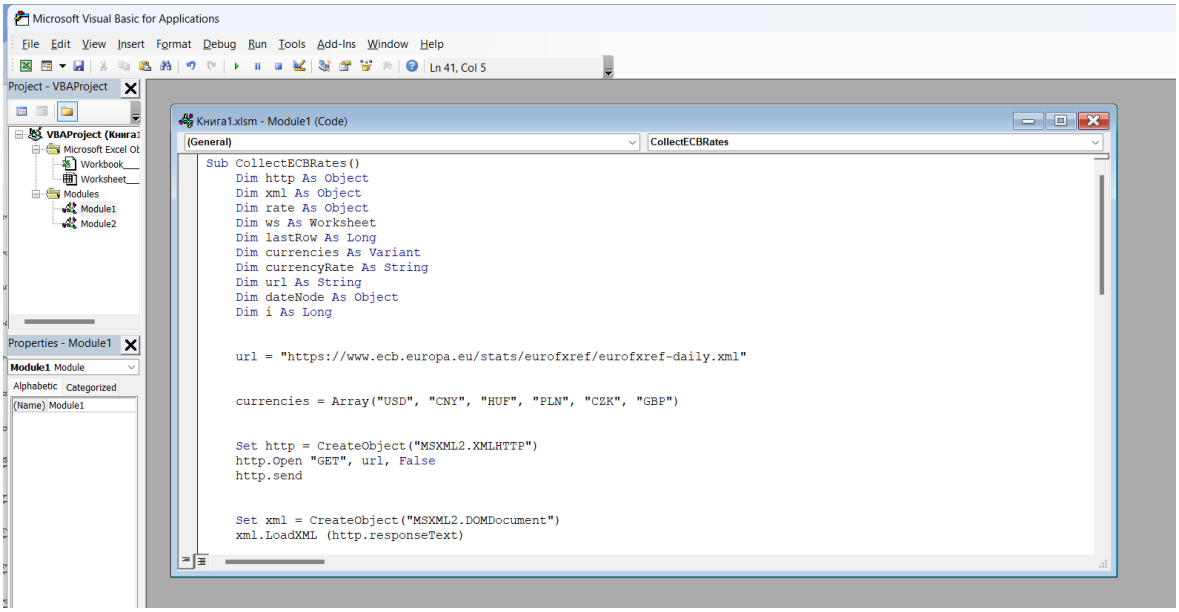
4.2.2 Alternatívny prístup pomocou Excel VBA a Plánovača úloh

Alternatívnym prístupom je využitie prostredia MS Excel s podporou programovania makier vo VBA . V rámci tohto prístupu bolo implementované makro, ktoré získava a ukladá aktuálne dáta valutových kurzov priamo do Excel tabuľky, pričom následne je spúšťané v pravidelných intervaloch pomocou plánovača úloh systému Windows.

Používanie VBA bolo zvolené ako jeden z počiatočných prístupov, pretože technológia je zabudovaná do programu Microsoft Excel a nevyžaduje inštaláciu ďalších knižníc alebo služieb.

Vyvinuté makro s názvom CollectECBRates zhromažďuje údaje vo formáte XML z webovej stránky ECB a automaticky ich vkladá do excelovskej tabuľky. Medzi hlavné fázy činnosti makra patria:

1. Odoslanie HTTP požiadavky na server ECB na získanie údajov vo formáte XML.
2. Získanie informácií o výmennom kurze pre konkrétnu skupinu mien: USD, CNY, HUF, PLN, CZK, GBP.
3. Spracovanie odpovede zo servera a automatizované vloženie dát do pripraveného Excel súboru, vrátane zaznamenania dátumu, aby sa zabezpečila úplnosť a konzistencia uchovávaných údajov.



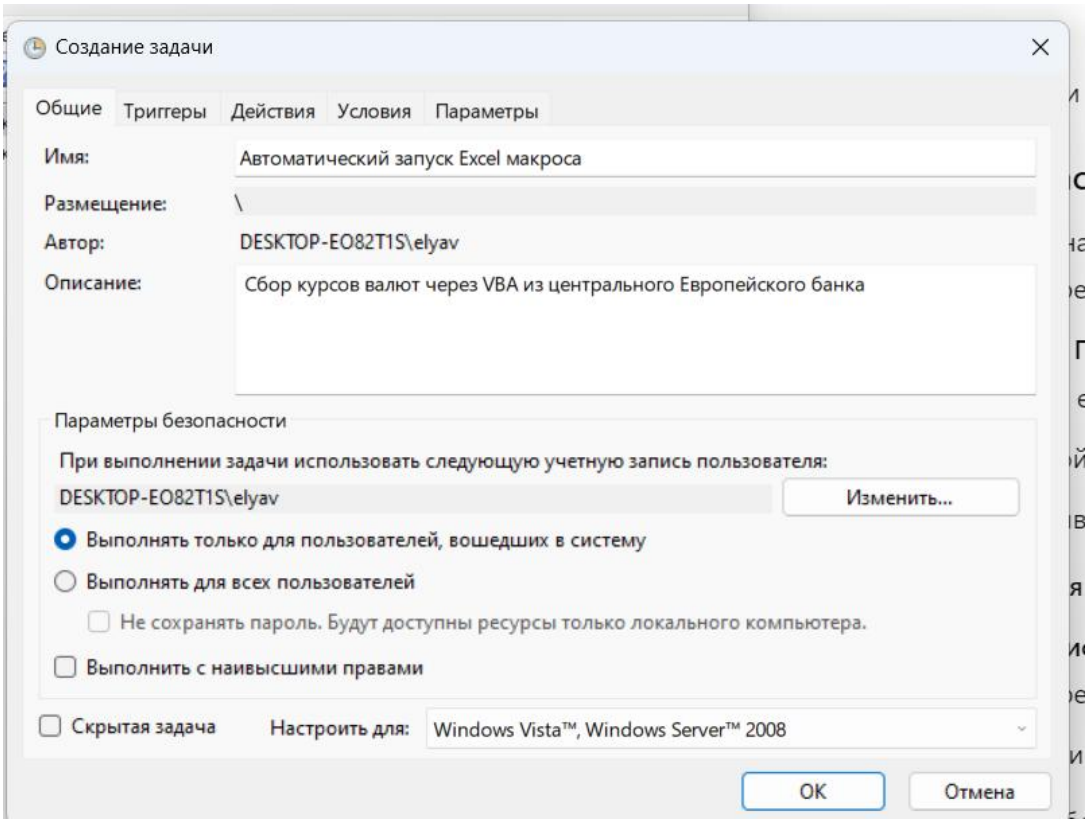
Obrázok 30: VBA editor v Exceli s makrom na zber kurzov
Zdroj: vlastné spracovanie

Pre automatizáciu spúšťania VBA makra bol využitý **Plánovač úloh Window**

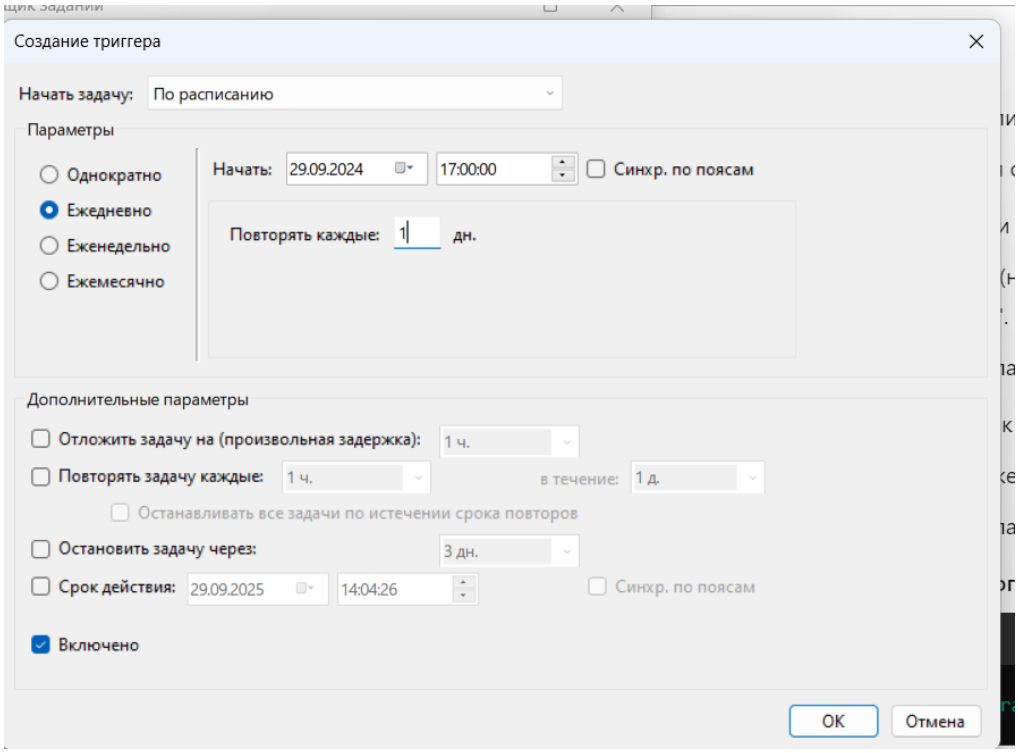
1. **Vytvorenie úlohy:** Bola vytvorená nová naplánovaná úloha s názvom *Automatický zber dát*, ako je znázornené na Obr. 27. Úloha je spustená pod konkrétnym používateľským účtom a má povolené vykonávanie aj v neprítomnosti používateľa.

2. **Spúšťanie úlohy:** V časti *Triggery* bolo nastavené **denné vykonávanie úlohy o 17:00**, so zapnutou možnosťou opakovania každý deň (Obr. 31).

3. **Akcia úlohy:** V časti *Akcie* je definované spustenie konkrétneho súboru .xlsm s implementovaným VBA makrom. Tento súbor sa po otvorení automaticky spustí, vykoná požadovaný zber dát a uloží výsledky do preddefinovaného hárku.



Obrázok 31: Vytvorenie úlohy v Plánovači úloh
Zdroj: vlastné spracovanie



Obrázok 32: Nastavenie plánovania úlohy na spustenie každý deň
Zdroj: vlastné spracovanie

4.3 Analýza príprava dát

Pred nasadením konkrétnych modelov do aplikácie bolo nevyhnutné vykonať testovanie rôznych modelov a vyhodnotiť ich výkonnosť pomocou vhodných metrík. Testovanie prebiehalo na historických dátach vybraných výmenných kurzov voči euru za obdobie od januára 2024.

4.3.1 Exploratívna analýza dát

Na účely testovania a ladenia predikčných modelov sme v počiatočnej fáze využívali **Excel súbor**, ktorý bol vytvorený pomocou **zberu dát cez VBA makrá** (viac v kapitole 4.2.2). Tento súbor obsahuje denné výmenné kurzy voči euru pre meny:

- USD – americký dolár
- CNY – čínsky jüan
- HUF – maďarský forint
- PLN – poľský zlotý
- CZK – česká koruna
- GBP – britská libra

Riešenie problému chýbajúcich údajov

Ako bolo uvedené v časti 3.1, údaje získavané z Európskej centrálnej banky obsahujú výmenné kurzy len za pracovné dni. Z toho dôvodu v dátovom súbore prirodzene chýbajú hodnoty za víkendy a sviatky. Na Obr. 33 je znázornená pôvodná časová os s chýbajúcimi hodnotami:

```
data.head(10)
```

	Date	USD	CNY	HUF	PLN	CZK	GBP
0	2024-01-02	1.0956	7.8264	382.10	4.3708	24.687	0.86645
1	2024-01-03	1.0919	7.8057	380.75	4.3638	24.675	0.86470
2	2024-01-04	1.0953	7.8330	378.85	4.3460	24.652	0.86278
3	2024-01-05	1.0921	7.8130	378.23	4.3568	24.616	0.86210
4	2024-01-08	1.0946	7.8397	377.65	4.3465	24.488	0.86150
5	2024-01-09	1.0940	7.8381	378.83	4.3448	24.594	0.85938
6	2024-01-10	1.0946	7.8476	378.35	4.3410	24.562	0.86023
7	2024-01-11	1.0987	7.8649	378.83	4.3490	24.659	0.86145
8	2024-01-12	1.0942	7.8451	379.35	4.3628	24.689	0.85950
9	2024-01-15	1.0945	7.8529	379.68	4.3625	24.714	0.86075

Obrázok 33: Zobrazenie datasetu s chýbajúcimi údajmi
Zdroj: vlastné spracovanie – Jupyter lab

Na vyplnenie chýbajúcich hodnôt boli zvážené nasledovné prístupy:

- **Forward-filling (doplňanie predchádzajúcou hodnotou):** Táto metóda predpokladá, že počas dní bez aktualizácie kurzov sa hodnota nemení. Pre každý deň bez hodnoty sa teda použije posledná známa hodnota kurzu.
- **Lineárna interpolácia:** Táto metóda vypočíta chýbajúce hodnoty na základe priemeru medzi známymi hodnotami pred a po výpadku. Je vhodná v prípade, že očakávame plynulé zmeny kurzu bez náhlych výkyvov.

Bol zvolený prístup **forward-filling**, ktorý sa ukázal ako najvhodnejší vzhľadom na špecifikum poskytovaných údajov. ECB totiž nemení kurz počas dní bez publikácie nových údajov, čo zodpovedá predpokladu nemennosti a zároveň zabezpečuje kompatibilitu s predikčnými modelmi, nakoľko väčšina modelov predpokladá, že vstupné dáta majú konzistentnú časovú frekvenciu – t. j. medzi jednotlivými hodnotami je rovnaký časový odstup.

Transformácia typu údajov. Knižnica pandas v prostredí Python pri načítaní údajov predvolene interpretuje stĺpce ako textové reťazce (`string`), vrátane tých, ktoré predstavujú dátum. Na to, aby bolo možné pracovať s časovým radom (napr. triedenie, filtrovanie podľa dátumu, výpočty období, vizualizácia a predikcia), je potrebné transformovať príslušný stĺpec na typ `datetime`. Proces doplnenia chýbajúcich dát a ich následnej transformácie je znázornený na Obr. 34:

```
data['Date'] = pd.to_datetime(data['Date']) # transformacia dat

# Создание полного диапазона дат (включая выходные и праздники)
full_date_range = pd.date_range(start=data['Date'].min(), end=data['Date'].max(), freq='D')

# Приведение данных к полному диапазону с заполнением пропущенных значений
data = data.set_index('Date').reindex(full_date_range).rename_axis('Date').reset_index()

# Заполнение пропущенных значений последним известным значением
data.fillna(method='ffill', inplace=True)

# Проверка на наличие пропусков
print("Пропущенные значения после обработки:\n", data.isnull().sum())
```

```
Пропущенные значения после обработки:
 Date    0
USD     0
CNY     0
HUF     0
PLN     0
CZK     0
GBP     0
dtype: int64
```

Obrázok 34: Transformácia dát a doplnenie chýbajúcich hodnôt
Zdroj: vlastné spracovanie – Jupyter lab

Doplnený časový rad vyzerá nasledovne:

```
data.head(15)
```

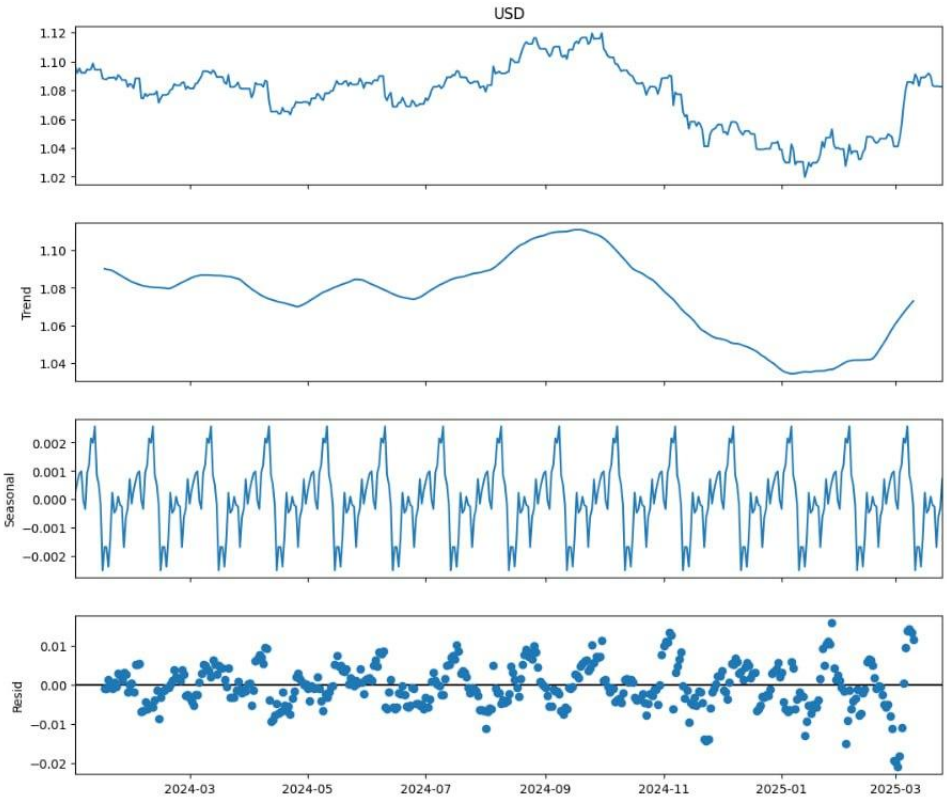
	Date	USD	CNY	HUF	PLN	CZK	GBP
0	2024-01-02	1.0956	7.8264	382.10	4.3708	24.687	0.86645
1	2024-01-03	1.0919	7.8057	380.75	4.3638	24.675	0.86470
2	2024-01-04	1.0953	7.8330	378.85	4.3460	24.652	0.86278
3	2024-01-05	1.0921	7.8130	378.23	4.3568	24.616	0.86210
4	2024-01-06	1.0921	7.8130	378.23	4.3568	24.616	0.86210
5	2024-01-07	1.0921	7.8130	378.23	4.3568	24.616	0.86210
6	2024-01-08	1.0946	7.8397	377.65	4.3465	24.488	0.86150
7	2024-01-09	1.0940	7.8381	378.83	4.3448	24.594	0.85938
8	2024-01-10	1.0946	7.8476	378.35	4.3410	24.562	0.86023
9	2024-01-11	1.0987	7.8649	378.83	4.3490	24.659	0.86145
10	2024-01-12	1.0942	7.8451	379.35	4.3628	24.689	0.85950
11	2024-01-13	1.0942	7.8451	379.35	4.3628	24.689	0.85950
12	2024-01-14	1.0942	7.8451	379.35	4.3628	24.689	0.85950
13	2024-01-15	1.0945	7.8529	379.68	4.3625	24.714	0.86075
14	2024-01-16	1.0882	7.8237	379.36	4.3870	24.710	0.86078

Obrázok 35: Doplnený časový rad
Zdroj: vlastné spracovanie – Jupyter lab

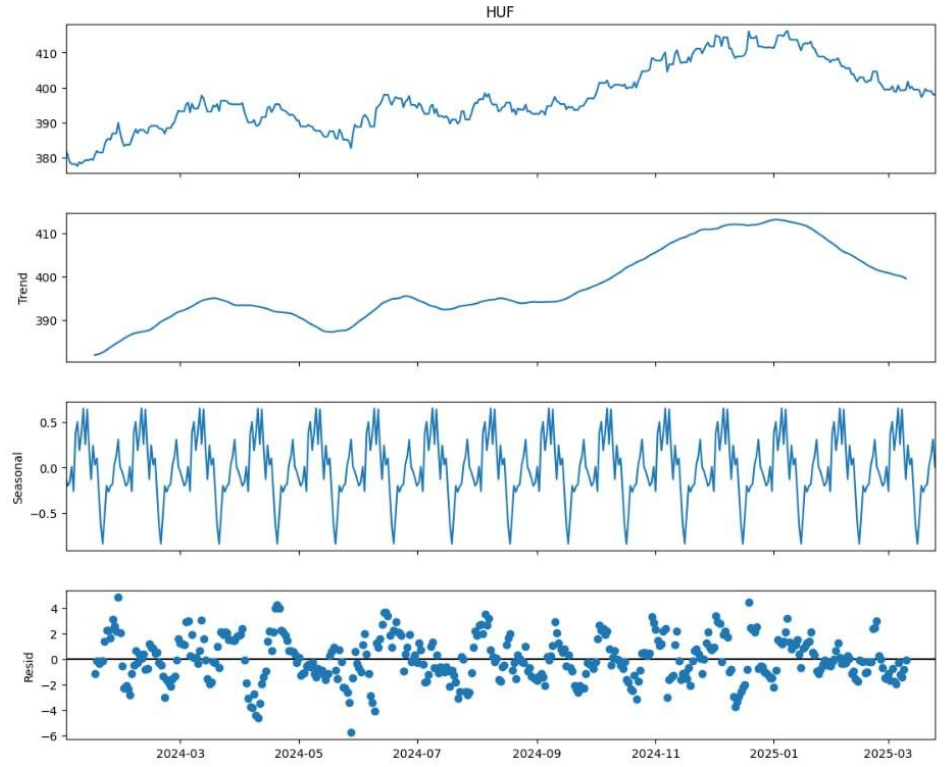
:Pre lepšie pochopenie správania sa výmenných kurzov v čase sme pristúpili k dekompozícii časového radu na 3 základné zložky pomocou aditívneho modelu.
Na tento účel bola využitá funkcia `seasonal_decompose` zo štandardnej knižnice `statsmodels`, konkrétne z modulu `tsa.seasonal`. Výsledné grafy jednotlivých komponentov boli vizualizované pomocou knižnice `matplotlib`.

Dekompozícia časových radov pomocou aditívneho modelu umožnila identifikovať štruktúru vývoja jednotlivých výmenných kurzov voči euru. V každom prípade boli analyzované tri hlavné zložky: **trend**, **sezónna zložka** a **reziduálna zložka:**

Na základe dekompozície časových radov (Obr. 36-37) možno pozorovať rozdiely v správaní jednotlivých mien. GBP a USD vykazujú najnižšiu volatilitu a len miernu sezónnosť, čo ich predurčuje na použitie jednoduchších modelov ako lineárna regresia alebo ARIMA.

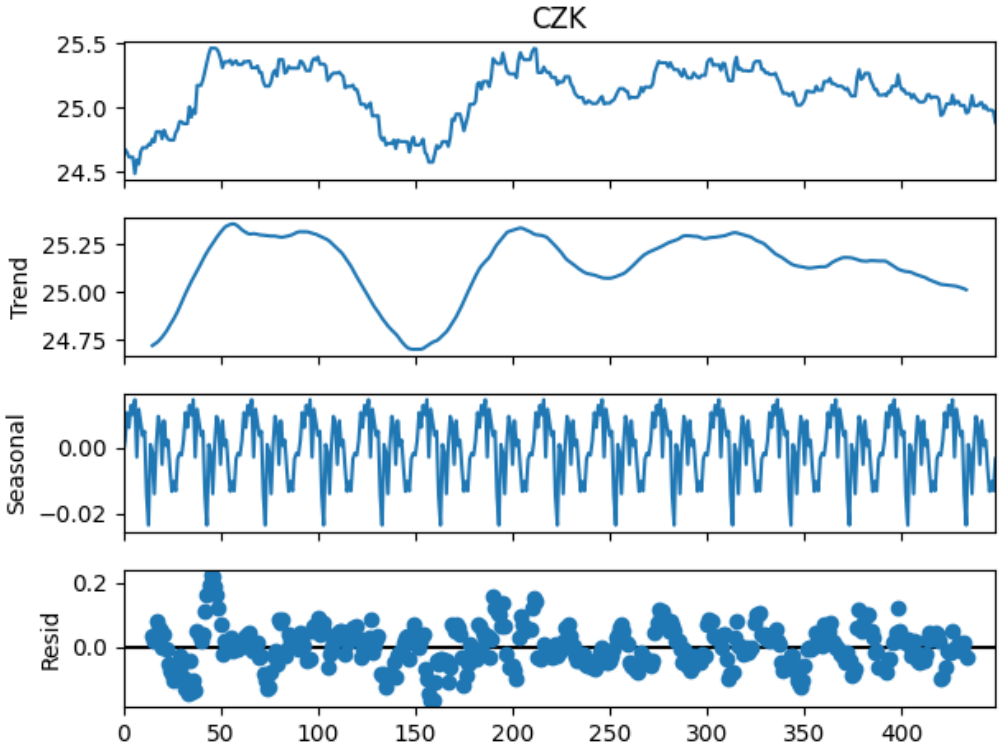


Obrázok 36: Dekompozícia časového radu pre kurz EUR/USD
Zdroj: vlastné spracovanie – Jupyter Lab



Obrázok 37: Dekompozícia časového radu pre kurz EUR/HUF
Zdroj: vlastné spracovanie – Jupyter Lab

PLN a **CZK (**Obr. 38) majú výraznejšiu sezónnosť a mierne nestabilný trend, čo si vyžaduje flexibilnejšie prístupy ako SARIMA alebo Prophet. Vyššia reziduálna variabilita poukazuje na potrebu robustnejších modelov schopných zvládať šum (napr. LSTM). **CNY** sa vyznačuje výraznými nelinearitami a potenciálnymi šokmi, čím sa stáva vhodným kandidátom pre modely s citlivosťou na štrukturálne zmeny, ako je Prophet alebo TimeGPT. Najvyššiu volatilitu vykazuje **HUF**, pri ktorom sa odporúča využitie zložitejších modelov, napríklad neurónových sietí.



Obrázok 38: Dekompozícia časového radu pre kurz EUR/CZK
Zdroj: vlastné spracovanie – Jupyter Lab

4.3.2 Predpracovanie dát pre modelovanie

Po základnom čistení a úprave datasetu nasleduje ďalšia fáza predspracovania údajov, ktorá je závislá od konkrétneho zvoleného predikčného modelu. Rôzne modely majú rozdielne požiadavky na vstupné dáta:

- Pre lineárnu regresiu je postačujúce, aby dáta boli vo forme dvojrozmerného poľa, kde každý riadok reprezentuje jednu vzorku s príslušnými vstupnými hodnotami.
- Model LSTM vyžaduje trojrozmerný tvar vstupných údajov, ktorý zahŕňa počet vzoriek, počet časových krokov a počet znakov.
- Model Prophet dokáže pracovať s časovým radom vo forme dátumovej postupnosti s pridelenými hodnotami bez potreby komplexnej transformácie.

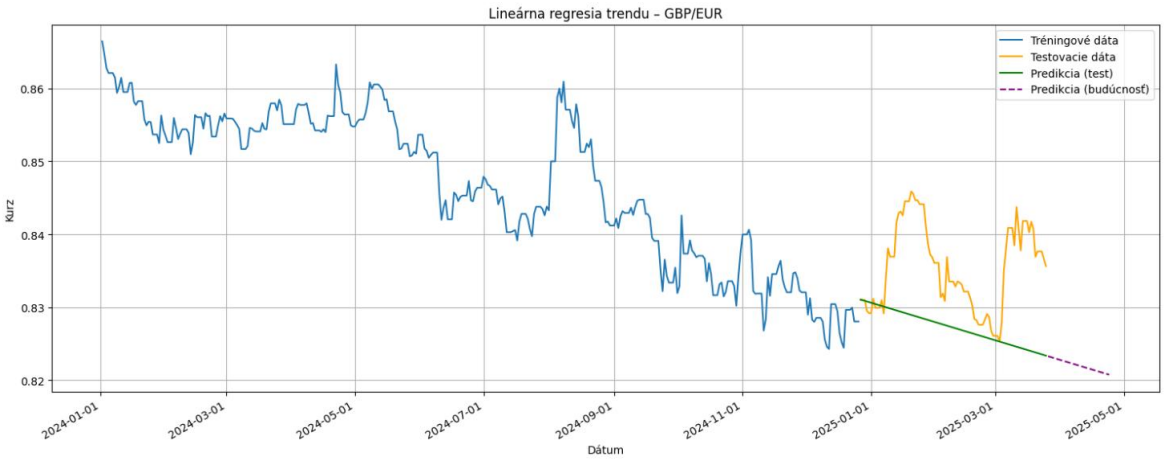
4.4 Predikčné modelovanie

Každý model bol aplikovaný na rovnaký dataset, ktorý bol rozdelený na tréningovú a testovaciu množinu v pomere 80:20. Predikcia bola následne porovnaná s reálnymi hodnotami kurzu pre testovaciu množinu. Použité vzorce boli detailne opísané v kapitole 3.3.

Všetky predikčné modely implementované v rámci tejto práce boli navrhnuté ako samostatné **funkcie** s cieľom zabezpečiť opätovnú použiteľnosť a flexibilitu pri testovaní rôznych parametrov.

4.4.1 Lineárna regresia

Model lineárnej regresie bol v rámci tejto práce použitý ako **základný benchmark** pre porovnanie s pokročilejšími predikčnými metódami. Konkrétne ide o **model lineárneho trendu**, ktorý predikuje výmenný kurz ako lineárnu funkciu času



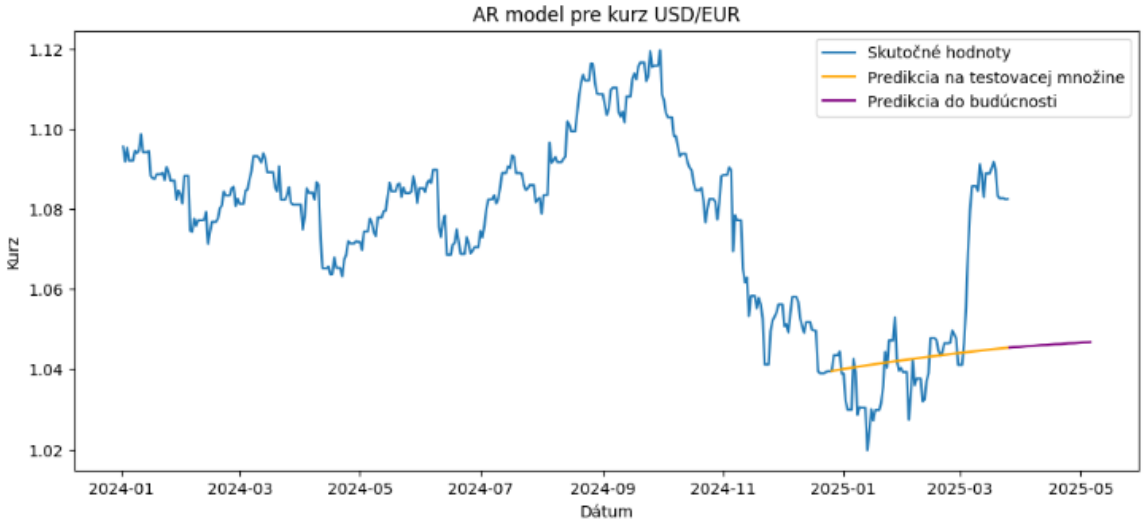
Obrázok 39:Graf predikcie "lineárna regresia pre GBP/EUR"
Zdroj: vlastné spracovanie – Jupyter Lab

Na základe výsledkov vizualizácie je zrejmé, že model lineárnej regresie nedokáže zachytiť komplexnú dynamiku vývoja výmenného kurzu. Predikcia na testovacej množine (zelená čiara) aj predikcia do budúcnosti (fialová prerušovaná čiara) vykazujú len jednoduchý lineárny trend, ktorý nereflektuje reálne výkyvy a prudké zmeny v historických údajoch. Model tak úplne zlyháva pri identifikovaní vzorov v časovom rade, čo potvrdzuje jeho obmedzenú vhodnosť pre tento typ dát.

4.4.2 Autoregresné modely ARIMA

V rámci testovania predikčných prístupov bol najskôr aplikovaný autoregresný model (AR). Tento model vychádza zo základnej myšlienky, že historické hodnoty

výmenných kurzov môžu poskytovať informáciu o ich budúcom vývoji. Na Obr. 40 je znázornený výstup AR modelu pre kurz USD/EUR, kde je zreteľne viditeľná predikcia v rámci testovacej množiny aj následná extrapolácia do budúcnosti.



Obrázok 40: Autoregresný model AR(5) pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Pred aplikáciou ARIMA modelu bola vykonaná analýza stacionarity pomocou ADF testov. Testy preukázali, že časový rad nie je plne stacionárny, čo by naznačovalo potrebu diferenciácie.

```
from statsmodels.tsa.stattools import adfuller

# ADF test
result = adfuller(series)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
for key, value in result[4].items():
    print('Critical Values:')
    print(f'   {key}, {value}')
```

```
ADF Statistic: -1.838279
p-value: 0.361629
Critical Values:
   1%, -3.4451978474132234
Critical Values:
   5%, -2.8680864144212057
Critical Values:
   10%, -2.5702569996789792
```

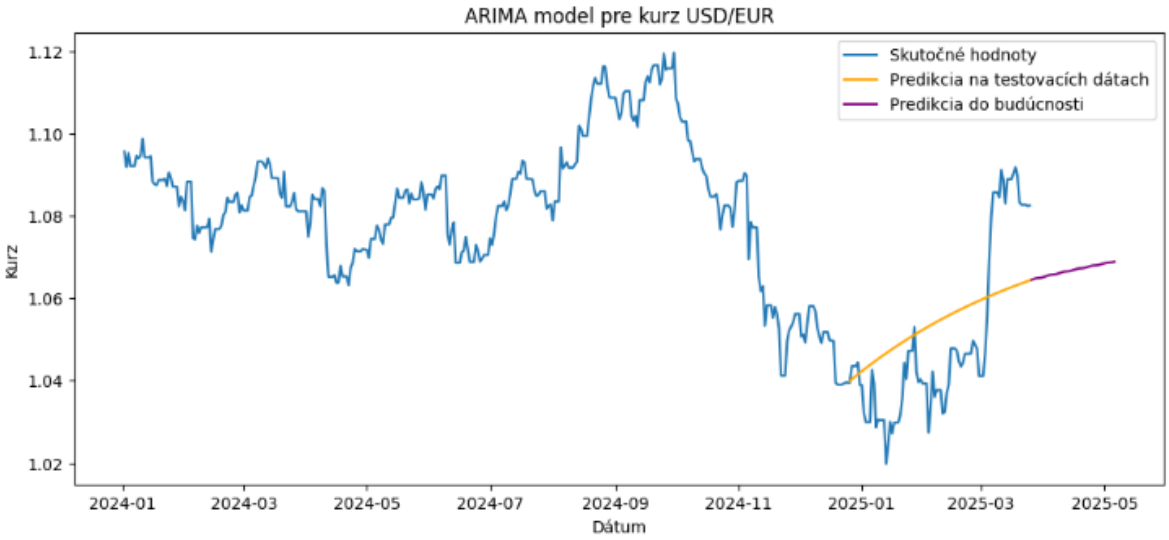
Obrázok 41: ADF test pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Pri implementácii modelu ARIMA bola venovaná osobitná pozornosť optimalizácii parametrov **(p, d, q)** na základe výpočtu informačných kritérií, predovšetkým **AIC (Akaike Information Criterion)**, ktorý pomáha identifikovať model s najlepším kompromisom medzi presnosťou predikcie a zložitosťou modelu.

```
ARIMA(0, 0, 0) MSE=0.001510
ARIMA(0, 0, 1) MSE=0.001492
ARIMA(0, 0, 2) MSE=0.001483
ARIMA(0, 1, 0) MSE=0.000535
ARIMA(0, 1, 1) MSE=0.000535
ARIMA(0, 1, 2) MSE=0.000536
ARIMA(0, 2, 0) MSE=0.000535
ARIMA(0, 2, 1) MSE=0.002114
ARIMA(0, 2, 2) MSE=0.002800
ARIMA(1, 0, 0) MSE=0.000292
ARIMA(1, 0, 1) MSE=0.000275
ARIMA(1, 0, 2) MSE=0.000288
ARIMA(1, 1, 0) MSE=0.000535
ARIMA(1, 1, 1) MSE=0.000535
ARIMA(1, 1, 2) MSE=0.000536
ARIMA(1, 2, 0) MSE=0.000428
ARIMA(1, 2, 1) MSE=0.002426
ARIMA(1, 2, 2) MSE=0.002946
ARIMA(2, 0, 0) MSE=0.000275
ARIMA(2, 0, 1) MSE=0.000292
ARIMA(2, 0, 2) MSE=0.000280
ARIMA(2, 1, 0) MSE=0.000535
ARIMA(2, 1, 1) MSE=0.000535
ARIMA(2, 1, 2) MSE=0.000535
ARIMA(2, 2, 0) MSE=0.000356
ARIMA(2, 2, 1) MSE=0.003009
ARIMA(2, 2, 2) MSE=0.003179

Najlepšie parametre ARIMA: (2, 0, 0) s MSE=0.000275
```

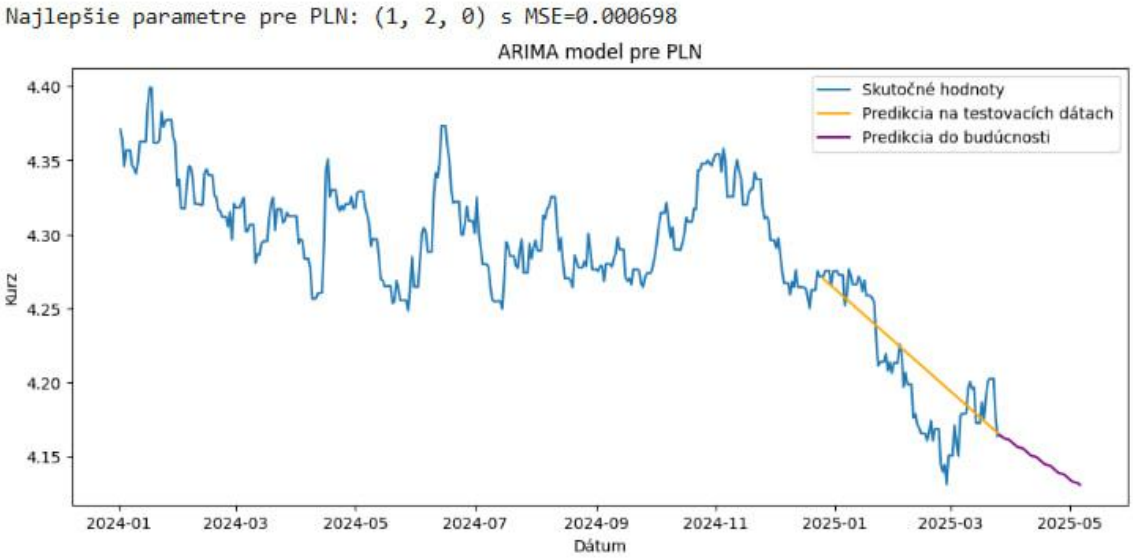
Obrázok 42: Optimalizácia parametrov modelu - výstup pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Na základe vykonanej optimalizácie pre časový rad USD/EUR bolo ako najvhodnejšie identifikované nastavenie ARIMA (2, 0, 0). To znamená, že model zahŕňa dve predchádzajúce hodnoty bez potreby diferenciácie (d = 0). V prípade menových kurzov, ktoré sa vyznačujú nízkou trendovosťou, ale vysokou volatilitou okolo strednej hodnoty, môže dôjsť k tomu, že pôvodný nediferencovaný rad zachováva prirodzené fluktuácie a závislosti v dátach lepšie, ako príliš "vyhladený" stacionárny rad. Pri diferenciácii na d=1 sa časový rad môže správať ako náhodná prechádzka (random walk), ktorá v tomto prípade neposkytuje pridanú hodnotu pre predikciu (Al-Shboul, 2020). Podobné zistenia prezentovali aj Meese a Rogoff (1983), podľa ktorých pri predikcii výmenných kurzov jednoduchšie modely niekedy poskytujú porovnateľné, ak nie lepšie výsledky ako zložitejšie špecifikácie s vyššou úrovňou diferenciácie.



Obrázok 43: Graf predikcie ARIMA modelu pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Napriek správnej technickej implementácii, výsledky autoregresného modelu ukázali, že pre predikciu výmenných kurzov nie je tento prístup dostatočne presný. Dôvodom je najmä nízka predikčná schopnosť modelu pri aplikácii na dáta s vysokou mierou volatility a výrazným vplyvom externých faktorov, ktoré model AR nedokáže zachytiť. Model má tendenciu generovať príliš hladké predikcie, ktoré nereflektujú reálnu dynamiku trhu.



Obrázok 44: Graf predikcie ARIMA modelu pre PLN/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

4.4.3 Modely hlbokého učenia – RNN a LSTM

Vzhľadom na špecifiká architektúry rekurentných neurónových sietí, použité modely Simple RNN a LSTM vyžadovali špecifickú predprípravu vstupných údajov, aby sa zabezpečila ich kompatibilita s požiadavkami modelov a optimalizoval sa proces učenia.

Pre oba modely bola aplikovaná metóda Min-Max škálovania:

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(series)
```

Pred použitím modelov je potrebné upraviť vstupné dáta do trojrozmerného tvaru s rozmermi [počet vzoriek, počet časových krokov, počet vstupných premenných]. Tento krok zabezpečuje nasledujúci príkaz:

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

Konkrétne:

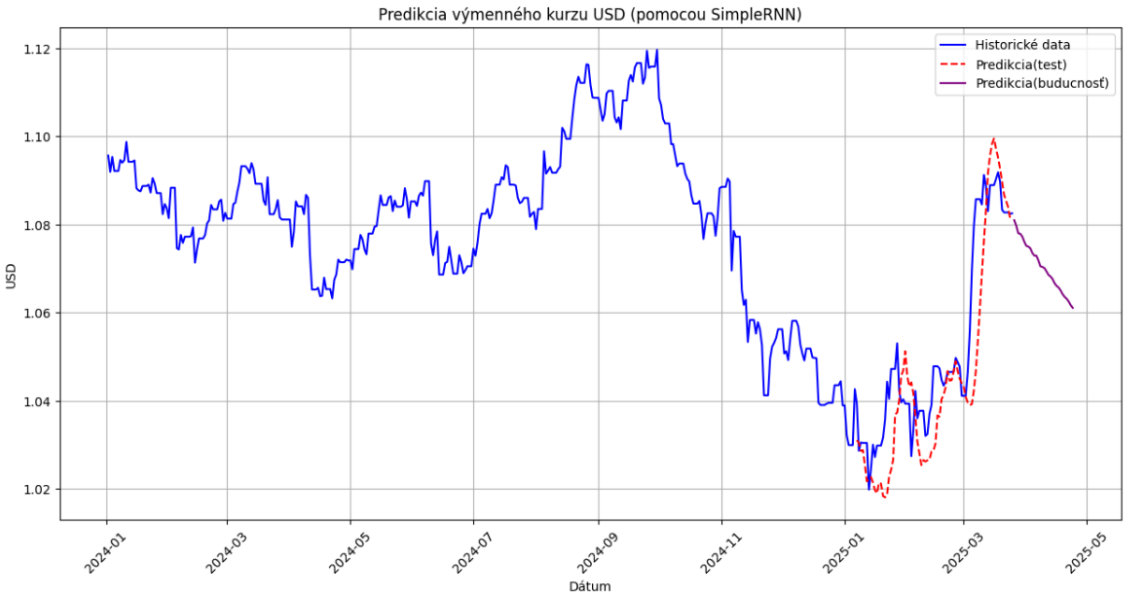
- X_train.shape[0] → počet vzoriek (samples)
- X_train.shape[1] → počet časových krokov v okne (look_back = 30)
- 1 → počet znakov (features), keďže predikujeme len jednu veličinu — výmenný kurz.

Model Simple RNN

Model Simple RNN využíva rekurentné vrstvy na zachytenie sekvenčných vzorcov vo vývoji výmenného kurzu. Pri tréningu modelu bolo použité **posuvné okno (sliding window)** s dĺžkou 60 dní (look_back = 60), v rámci ktorého sa každý vstupný vzor vytváral z posledných 60 zaznamenaných hodnôt výmenného kurzu. Na základe týchto sekvencií model predikoval budúci vývoj kurzu v horizonte 30 dní:

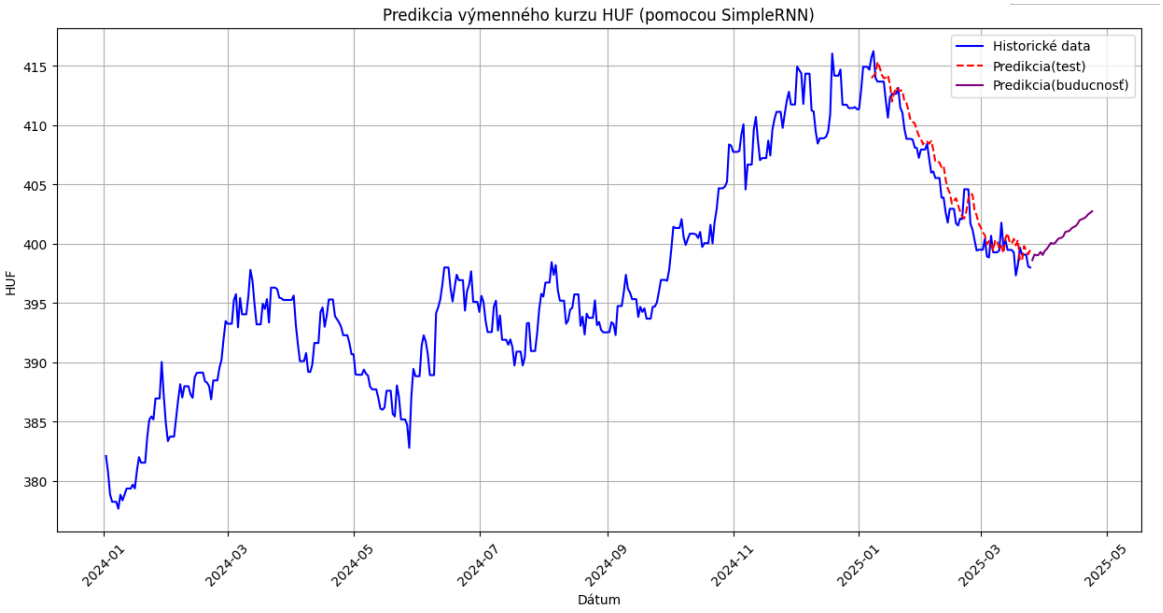
```
forecast = simpleRNN_forecast(data, 'USD', look_back=60,
forecast_steps=30)
```

Model Simple RNN preukázal veľmi dobrú schopnosť zachytiť vývoj výmenného kurzu USD (Obr.45). Predikcia na testovacej množine (červená čiara) tesne kopíruje skutočný priebeh kurzu, čo naznačuje, že model správne identifikoval vzory v historických dátach. Aj predikcia na budúce obdobie (fialová čiara) zachováva logický trend a pokračovanie dynamiky vývoja.



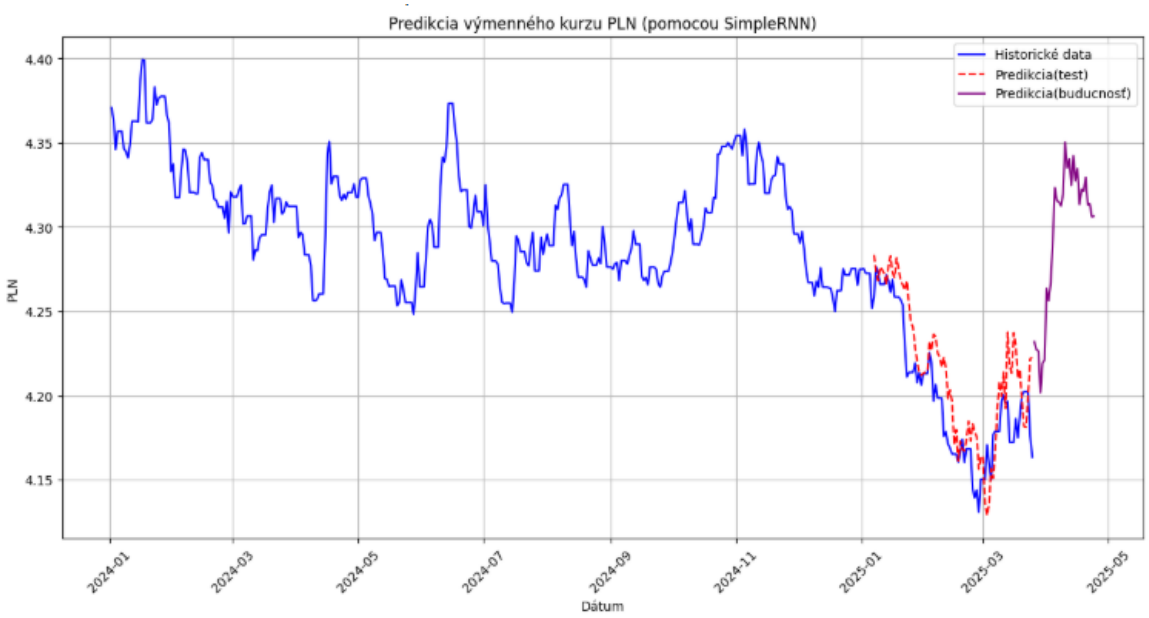
Obrázok 45: Graf predikcie modelu Simple RNN pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Maďarský forint je považovaný za jednu z menej stabilných mien v regióne, s vyššou náchylnosťou na výkyvy spôsobené menovou politikou, regionálnymi šokmi či špecifickými makroekonomickými faktormi. Cieľom bolo overiť, ako si model poradí s predikciou v podmienkach zvýšenej volatility.



Obrázok 46: Graf predikcie modelu Simple RNN pre HUF/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Výsledky ukazujú, že model dokázal **veľmi presne zachytiť zostupný trend** kurzov HUF a PLN v rámci testovacej množiny (červená prerušovaná čiara), vrátane niektorých menších korekcií**.**



Obrázok 47: Graf predikcie modelu Simple RNN pre PLN/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Model LSTM

Model LSTM bol zahrnutý do testovania ako rozšírenie Simple RNN modelu, a to vzhľadom na jeho schopnosť efektívnejšie pracovať s dlhodobejšími závislosťami v dátach. Simple RNN modely síce umožňujú spracovanie sekvenčných údajov, avšak pri dlhších sekvenciách sa stretávajú s problémom miznúcich gradientov, čo vedie k strate informácií zo vzdialenejších časových krokov.

Podľa teoretických poznatkov a dostupnej literatúry je model LSTM považovaný za vhodnejší pre úlohy predikcie časových radov, keďže jeho architektúra umožňuje uchovávanie dôležitých informácií z predchádzajúcich období a riadenie toku informácií prostredníctvom bránovej štruktúry.

Štruktúra siete:

- **Prvá LSTM vrstva:**

Obsahuje 50 neurónov a je nastavená s return_sequences=True, aby mohla výstup odovzdať do ďalšej LSTM vrstvy v sekvencii.

- **Dropout:**

Po prvej LSTM vrstve je aplikovaný dropout s mierou 0.2, čím sa redukuje riziko pretrénovania.

- **Druhá LSTM vrstva:**

Ďalších 50 neurónov spracúva výstup z predchádzajúcej vrstvy, pričom return_sequences=False ukončuje sekvenciu a pripravuje dáta pre plne prepojenú vrstvu.

- **Druhý Dropout:**

Opätovne použitý dropout s hodnotou 0.2 na zvýšenie generalizačnej schopnosti modelu.

- **Výstupná Dense vrstva:**

Lineárny neurón, ktorý generuje konečný predikovaný kurz.

Model pozostáva z dvoch vrstiev LSTM s 50 neurónmi, medzi ktorými je použitá regularizačná vrstva Dropout na zníženie rizika preučenia. Na záver je použitá plne prepojená vrstva Dense, ktorá produkuje jednu výstupnú hodnotu predikcie.

Tréning prebieha počas 50 epoch s veľkosťou batchu 32. Pre zhodnotenie výkonnosti modelu na testovacích dátach je použité aj validačné hodnotenie.

```
# 6. Создание модели LSTM
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(look_back, 1)))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')

# 7. Обучение модели
history = model.fit(
    X_train,
    y_train,
    epochs=50,
    batch_size=32,
    validation_data=(X_test, y_test),
    verbose=1
)
```

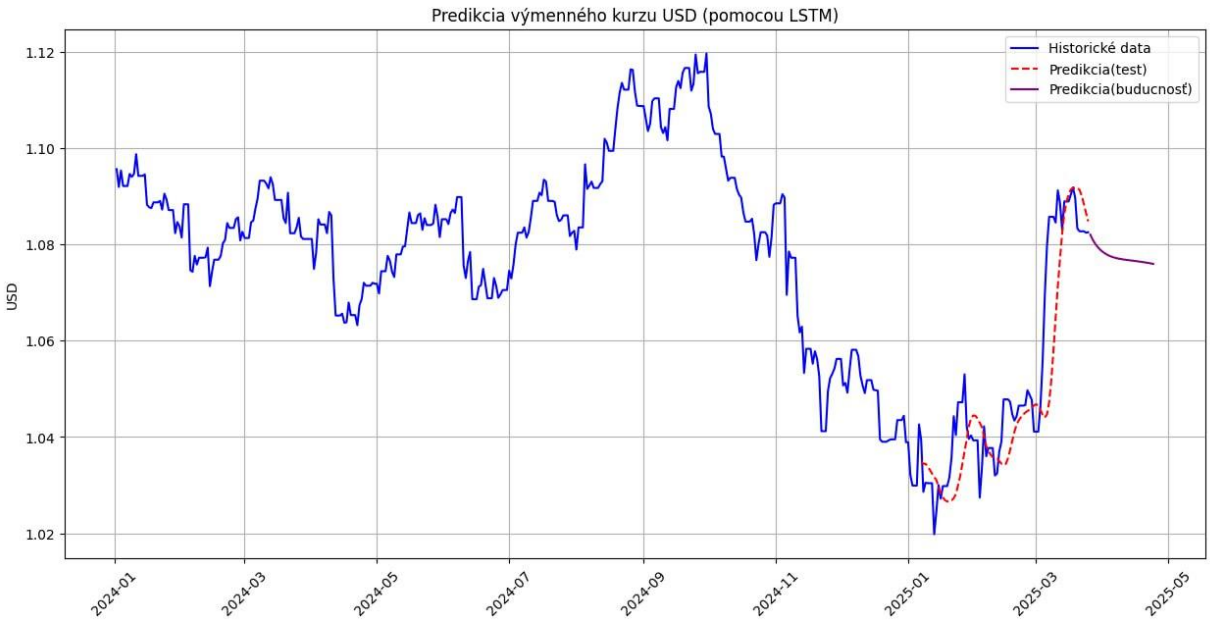
Obrázok 48: Vytváranie a tréning modelu LSTM
Zdroj: vlastné spracovanie – Jupyter Lab

Po natrénovaní modelu je vykonaná predikcia na testovacích dátach pre overenie kvality modelu a následne aj predikcia budúcich hodnôt na základe najnovších dostupných údajov.

Pre predikciu výmenného kurzu amerického dolára voči euru bola funkcia volaná nasledovne:

```
forecast = lstm_forecast(data, 'USD', look_back=60, forecast_steps=30)
```

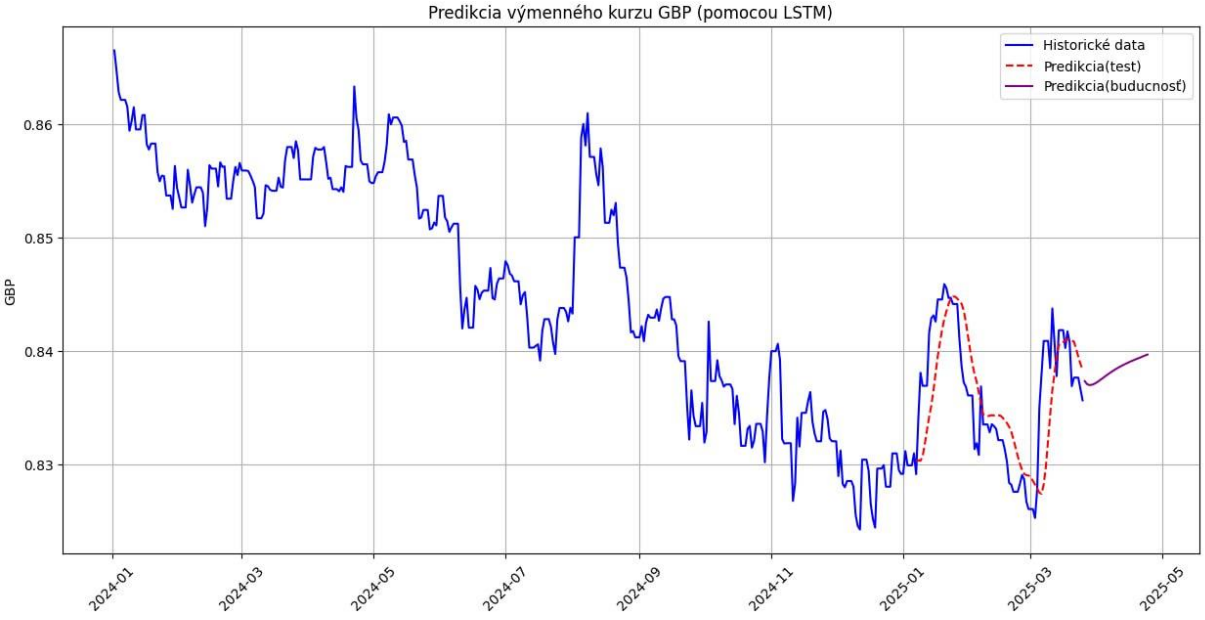
Na Obr. 49 je možné vidieť, že model predpokladá mierne klesajúci trend kurzu USD voči euru v predikcii na budúce obdobie. Tento vývoj je logický, nakoľko model vychádza z posledného vývoja a bez externých makroekonomických faktorov (ktoré nie sú súčasťou modelu) predpokladá návrat k priemeru, čo je pre neurónové siete bežné správanie.



Obrázok 49: Graf predikcie LSTM pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Na testovacích dátach model kopíroval reálny vývoj kurzu relatívne presne, najmä v obdobiach výrazného nárastu alebo poklesu.

Graficky je vidieť, že predikcia na testovacích dátach veľmi dobre kopíruje skutočný priebeh kurzu vrátane náhlych zmien smeru, čo potvrdzuje, že LSTM model je vhodný na predikcie menových časových radov. Rovnaký trend bol pozorovaný aj pri časovom rade GBP/EUR, kde model rovnako presne vystihol vývoj kurzu vrátane krátkodobých fluktuácií a bodov zlomu.

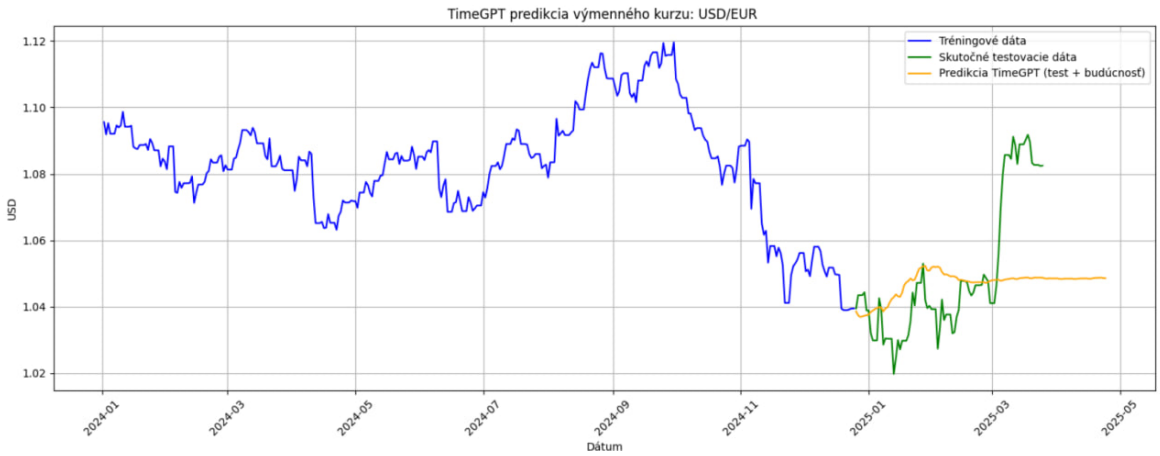


Obrázok 50: Graf predikcie LSTM pre GBP/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

4.4.4 Model Time-GPT

Model TimeGPT predstavuje najmodernejší prístup k predikcii časových radov, ktorý využíva architektúru tzv. transformerov. Na rozdiel od tradičných modelov neurónových sieti, ako sú Simple RNN alebo LSTM, ktoré spracúvajú dáta sekvenčne, transformerová architektúra umožňuje paralelné spracovanie celého časového radu.

Implementácia prebiehala prostredníctvom cloudovej API služby poskytovateľa **Nixtla**, čo umožnilo využitie predtrénovaného modelu bez potreby lokálneho trénovania na historických údajoch. Model bol konfigurovaný s parametrami optimalizovanými na predikciu krátkodobého vývoja kurzov, pričom osobitná pozornosť bola venovaná nastaveniu horizontu predikcie (forecast_steps) a počtu jemných doladení modelu (finetune_steps).



Obrázok 51: Graf predikcie modelu TimeGPT pre USD/EUR
Zdroj: vlastné spracovanie – Jupyter Lab

Na vizualizácii predikcie modelu TimeGPT je možné pozorovať, že model síce dokáže pomerne presne sledovať lokálne trendy v krátkodobom horizonte, avšak pri dlhodobejšej predikcii stráca schopnosť zachytiť komplexnejšie vzory vývoja výmenného kurzu. Predikovaná krivka (oranžová čiara) sa stabilizuje na približne konštantnej úrovni, čo naznačuje, že model nedokáže adekvátne reflektovať dynamické zmeny a dlhodobejšie výkyvy, ktoré sú prítomné v historických údajoch.

4.4.5 Model Prophet

Model Prophet okrem bodovej predikcie automaticky generuje aj interval spoľahlivosti na úrovni 95 %, ktorý poskytuje prehľad o možnom rozpätí predikovaných hodnôt. Tento interval zvyšuje interpretovateľnosť výsledkov a pomáha lepšie vyjadriť

neistotu predikcie. Príprava dát zahŕňa premenovanie stĺpcov do formátu vyžadovaného modelom Prophet, konkrétne ds (dátumová zložka) a y (cieľová premenná – výmenný kurz). Model zároveň umožňuje explicitne zohľadniť sezónne vplyvy a kalendárne udalosti, ako sú sviatky, ktoré boli v rámci tejto práce manuálne definované vzhľadom na ich potenciálny vplyv na vývoj výmenných kurzov.

Pri nastavovaní sezónnosti bola aktivovaná **ročná a týždenná sezónnosť**, ktoré reflektujú dlhodobejšie opakujúce sa vzory. Prophet ponúka aj možnosť zahrnúť **dennú sezónnosť**, no táto bola v našom prípade vypnutá (`daily_seasonality=False`), keďže pracujeme s dennými dátami, kde sa v rámci jedného dňa výmenný kurz nemení.

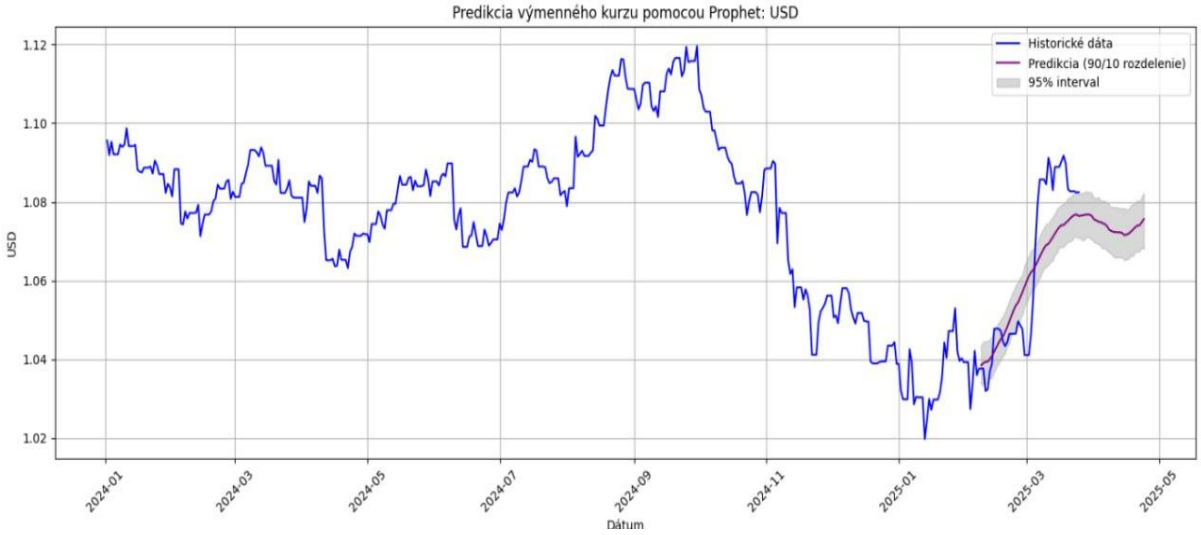
Prophet

```python
from prophet import Prophet
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

def prophet_forecast_full80(data, target_col, forecast_steps=30):
    """
    Prophet predikcia: model trénovaný len na 80 % dát, predikcia až od tej hranice do budúcnosti.
    Parametre:
    - data: DataFrame s indexom a kurzami
    - target_col: cieľový stĺpec (napr. 'USD')
    - forecast_steps: počet dní dopredu
    Výstup:
    - DataFrame s budúcou predikciou
    """
    # • Príprava dát
    df = data.reset_index()[['Date', target_col]].copy()
    df.rename(columns={'Date': 'ds', target_col: 'y'}, inplace=True)

    # • Rozdelenie 80/20 (len pre tréning)
    split_index = int(len(df) * 0.8)
    train_df = df.iloc[:split_index]
    full_df = df.copy()
    df_test = df.iloc[split_index:]

    # • Pridanie sviatkov
    holidays = pd.DataFrame({
        'holiday': 'sviatok',
        'ds': pd.to_datetime([
            '2024-01-01', '2024-04-01', '2024-05-01', '2024-05-08',
            '2024-07-05', '2024-08-29', '2024-09-01', '2024-09-15',
            '2024-11-01', '2024-12-24', '2024-12-25', '2024-12-26',
            '2025-01-01'
        ]),
        'lower_window': 0,
        'upper_window': 0
    })

    # • Tréning modelu len na 80 % dát
    model = Prophet(
        holidays=holidays,
        yearly_seasonality=True,
        weekly_seasonality=True,
        daily_seasonality=False
    )
    model.fit(train_df)
```

Obrázok 52: Konfigurácia modelu Prophet so sezónnymi komponentmi a sviatkami
Zdroj: vlastné spracovanie

Obrázok 53: Graf predikcie modelu Prophet pre USD/EUR (80:20)
Zdroj: vlastné spracovanie – Jupyter Lab

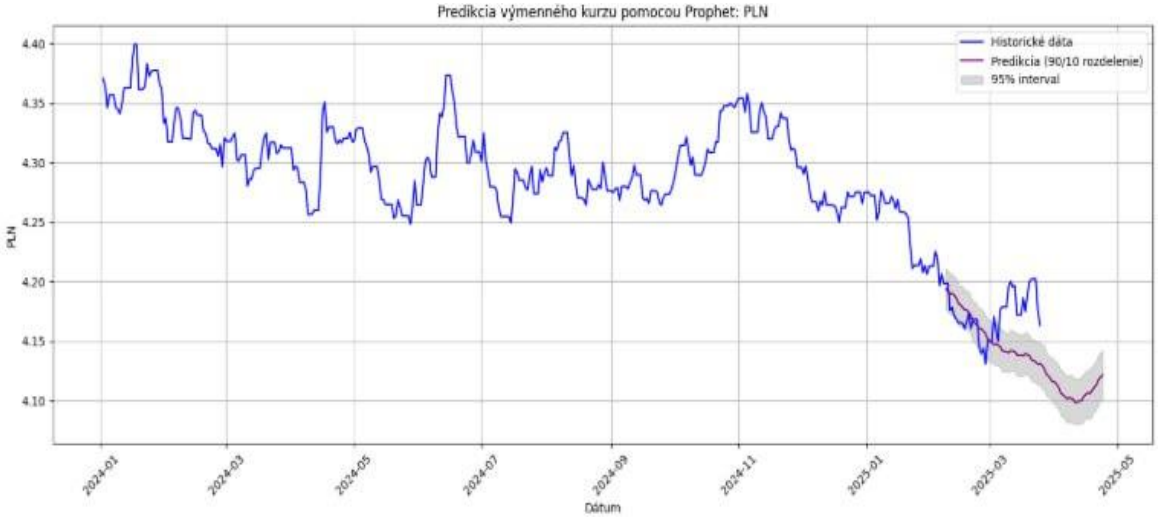
Pri pôvodnom rozdelení 80/20 model Prophet neposkytoval dostatočne presné výsledky pre niektoré časové rady, najmä pri vyššej volatilite údajov. Zvýšením podielu trénovacích dát na 90 % sa presnosť modelu výrazne zlepšila, keďže mal k dispozícii viac historických informácií na zachytenie trendov a sezónnych vzorov. Tento experiment potvrdil citlivosť Prophet na rozsah trénovacej množiny.



Obrázok 54:Graf predikcie modelu Prophet pre USD/EUR (90:10)
Zdroj: vlastné spracovanie – Jupyter Lab

Rovnaký efekt zlepšenia výkonnosti sme pozorovali aj pri ostatných analyzovaných menových pároch.

Obrázok 55: Graf predikcie modelu Prophet pre PLN/EUR (80:20)
Zdroj: vlastné spracovanie – Jupyter Lab



Obrázok 56: Graf predikcie modelu Prophet pre PLN/EUR (90:10)
Zdroj: vlastné spracovanie – Jupyter Lab

Výsledky modelu Prophet pre kurz PLN/EUR pri použití 90 % trénovacích dát ukazujú, že model je schopný zachytiť smerovanie vývoja v krátkodobom horizonte. Predikcia nasleduje existujúci trend a zachováva konzistentnosť s historickým vývojom, pričom interval spoľahlivosti odráža mieru neistoty. Na základe týchto výstupov možno konštatovať, že model Prophet je vhodný najmä pre krátkodobé predikcie.

4.4.6 Porovnanie výsledkov modelov a analýza presnosti

Presnosť predikcie sme hodnotili pomocou štandardných metrík: **MAE**, **RMSE** a **MSE.**

```
print(f"MAE: {mean_absolute_error(y_test_orig, test_predict):.4f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_orig, test_predict)):.4f}")
print(f"MSE: {mean_squared_error(y_test_orig, test_predict):.4f}")
```

Tabuľka 5: Porovnanie modelov podľa vybraných metrík presnosti (USD/EUR)
Zdroj: vlastné spracovanie

Model predikcie	MAE	RMSE	MSE
LSTM	0.0067	0.0097	0.0001
Simple RNN	0.0085	0.0111	0.0001
Prophet (80/20)	0.0419	0.0475	0.0023
Prophet (90/10)	0.0099	0.0117	0.0001
TimeGPT (Nixtla)	0.0147	0.0198	0.0004
Lineárna regresia	0.0307	0.0333	0.0011
AR	0.0141	0.0207	0.0004
ARIMA (2,0,0)	0.0148	0.0003	0.0166
ARIMA (2,1,0)	0.0635	0.0764	0.0058

Tabuľka 6: Porovnanie modelov podľa vybraných metrík presnosti (CNY/EUR)
Zdroj: vlastné spracovanie

Model predikcie	MAE	RMSE	MSE
LSTM	0.0390	0.0581	0.0034
Simple RNN	0.0480	0.0694	0.0048
Prophet (80/20)	0.1433	0.1853	0.0343
Prophet (90/10)	0.0955	0.1235	0.0152
TimeGPT (Nixtla)	0.0922	0.1308	0.0171
Lineárna regresia	0.1480	0.1530	0.0234
AR(5)	0.0959	0.1097	0.0120
ARIMA(2, 0, 2)	0.0971	0.1089	0.0119

Tabuľka 7: Porovnanie modelov podľa vybraných metrík presnosti (PLN/EUR)
Zdroj: vlastné spracovanie

Model predikcie	MAE	RMSE	MSE
LSTM	0.0277	0.0321	0.0010
Simple RNN	0.0183	0.0232	0.0005
Prophet (80/20)	0.0872	0.0935	0.0087
Prophet (90/10)	0.0272	0.0342	0.0012
TimeGPT (Nixtla)	0.0670	0.0798	0.0064
Lineárna regresia	0.0711	0.0067	0.0067
AR (5)	0.0878	0.1006	0.0101
ARIMA (1,2,0)	0.0218	0.0007	0.0264

Tabuľka 8: Porovnanie modelov podľa vybraných metrík presnosti (GBP/EUR)
Zdroj: vlastné spracovanie

Model predikcie	MAE	RMSE	MSE
LSTM	0.0025	0.0034	0.00001
Simple RNN	0.0020	0.0028	0.00001
Prophet (80/20)	0.0087	0.0106	0.0001
Prophet (90/10)	0.0215	0.0262	0.0007
TimeGPT (Nixtla)	0.0084	0.0103	0.0001
Lineárna regresia	0.0084	0.0104	0.0001
AR(5)	0.0057	0.0072	0.0001
ARIMA (1,0,2)	0.0050	0.0065	0.00001

Tabuľka 9: Porovnanie modelov podľa vybraných metrík presnosti (CZK/EUR)
Zdroj: vlastné spracovanie

Model predikcie	MAE	RMSE	MSE
LSTM	0.0452	0.0603	0.0036
Simple RNN	0.0375	0.0481	0.0023
Prophet (80/20)	1.0039	1.3357	1.1557
Prophet (90/10)	0.2116	0.2410	0.0581
TimeGPT (Nixtla)	0.2047	0.2369	0.0561
Lineárna regresia	0.1783	0.2036	0.0414
AR(5)	0.0942	0.1132	0.0128
ARIMA (1,0,0)	0.0588	0.0743	0.0055

Tabuľka 10: Porovnanie modelov podľa vybraných metrík presnosti (HUF/EUR)
Zdroj: vlastné spracovanie

Model predikcie	MAE	RMSE	MSE
LSTM	1.6560	1.9192	3.6832
Simple RNN	1.6597	1.8712	3.5015
Prophet (80/20)	15.2862	18.788	353.02
Prophet (90/10)	1.9565	2.7473	7.5475
TimeGPT (Nixtla)	5.5752	6.1798	38.1896
Lineárna regresia	7.5241	8.7750	77.0005
AR(5)	5.3457	6.4292	41.3343
ARIMA (2,0,1)	3.4447	4.0604	16.4869

Model Simple RNN dosiahol najnižšiu absolútnu chybu v prípade GBP/EUR, PLN/EUR a CZK/EUR, čím preukázal svoju schopnosť presne modelovať stabilnejšie alebo mierne volatilné menové rady.

Na druhej strane, model LSTM preukázal vyššiu robustnosť pri predikcii nestabilnejších kurzov, ako sú HUF/EUR a CNY/EUR, kde výrazne prekonal ostatné prístupy. Zaujímavým zistením je, že model ARIMA(1,0,2) dosiahol najnižšiu MAE pri predikcii USD/EUR (0.0050), čím prekonal aj zložitejšie neurónové siete. To poukazuje na to, že klasické štatistické prístupy môžu byť vhodné pri predikcii stabilných kurzov so silnou trendovou zložkou.

Model Prophet preukázal výraznú citlivosť na veľkosť trénovacej množiny – pri rozdelení 90/10 sa jeho presnosť zlepšila (napr. PLN/EUR z MAE = 0.0878 na MAE = 0.0272), avšak v prípade vysokej volatility (napr. HUF/EUR) výrazne zaostával za modelmi hlbokého učenia. Prophet sa tak javí ako vhodný najmä pre krátkodobé pre-dikcie v stabilnejších podmienkach.

Model ARIMA dosiahol najhoršie výsledky v prípade väčšiny menových párov – najmä pri HUF/EUR (MAE až 3.4447), čím potvrdil, že štandardné štatistické modely nie sú vhodné pre dynamické a nelineárne časové rady s výraznou volatilitou.

TimeGPT (Nixtla) dosiahol konzistentné výsledky, ktoré neboli najlepšie, ale v niektorých prípadoch konkuroval modelom Prophet. Jeho výhodou je rýchla implementácia bez potreby ladenia parametrov, avšak zatiaľ neprekonal výkonnosť klasických RNN modelov.

Výber modelov na nasadenie do aplikácie

Ako **benchmarkový základ** bola použitá **lineárna regresia**, ktorá síce poskytuje rýchle a stabilné riešenie, avšak vzhľadom na svoju jednoduchosť nedokáže zachytiť dynamiku a sezónnosť časových radov. Je však praktickým referenčným bodom pre porovnávanie komplexnejších modelov.

Hoci sa modely typu ARIMA bežne používajú pri analýze a predikcii časových radov, v prípade menových kurzov nie sú vhodné. Dôvodom je, že výmenné kurzy patria medzi finančné časové rady, ktoré sa vyznačujú nestacionaritou, zvýšenou volatilitou a často aj heteroskedasticitou. Tieto charakteristiky porušujú základné predpoklady ARIMA modelov, ako je stacionarita alebo konštantný rozptyl chýb (homoskedasticita).

Okrem týchto teoretických obmedzení je ARIMA model nepraktický z pohľadu implementácie v aplikácii, pretože vyžaduje manuálny výber a ladenie parametrov p, d, q,

čo znižuje jeho použiteľnosť v automatizovanom systéme. Na modelovanie meniacej sa volatility by síce bolo vhodné použiť ARCH alebo GARCH modely, ktoré sú špeciálne navrhnuté pre túto úlohu. Tie však neboli v tejto práci použité, keďže cieľom bolo predikovať budúce hodnoty výmenného kurzu samotného, nie jeho volatilitu.

Na základe komplexného porovnania presnosti predikcie výmenných kurzov na testovacej množine sme pre nasadenie do webovej aplikácie vybrali nasledovné modely:

- **Model neurónových sietí:** bolo rozhodnuté nasadiť do aplikácie len jeden model z dôvodu výpočtovej náročnosti a potreby optimalizácie produkčného riešenia. Keďže oba testované modely (LSTM a Simple RNN) preukázali dobrú presnosť, výber padol na LSTM, ktorý je vhodnejší pre zachytávanie dlhodobejších závislostí v časovom rade a zároveň dosahoval najlepšie výsledky pri volatilnejších menových pároch.
- **Prophet**: ako doplnkový model vhodný na krátkodobé predikcie s možnosťou vizualizácie trendov, sezónnosti a intervalov spoľahlivosti,
- **TimeGPT (Nixtla)**: ako moderný transformerový model, ktorý nevyžaduje manuálnu optimalizáciu parametrov a umožňuje predikciu aj bez priameho tréningu na lokálnych dátach.

4. 4. 7 Integrácia predikčných modelov do webovej aplikácie

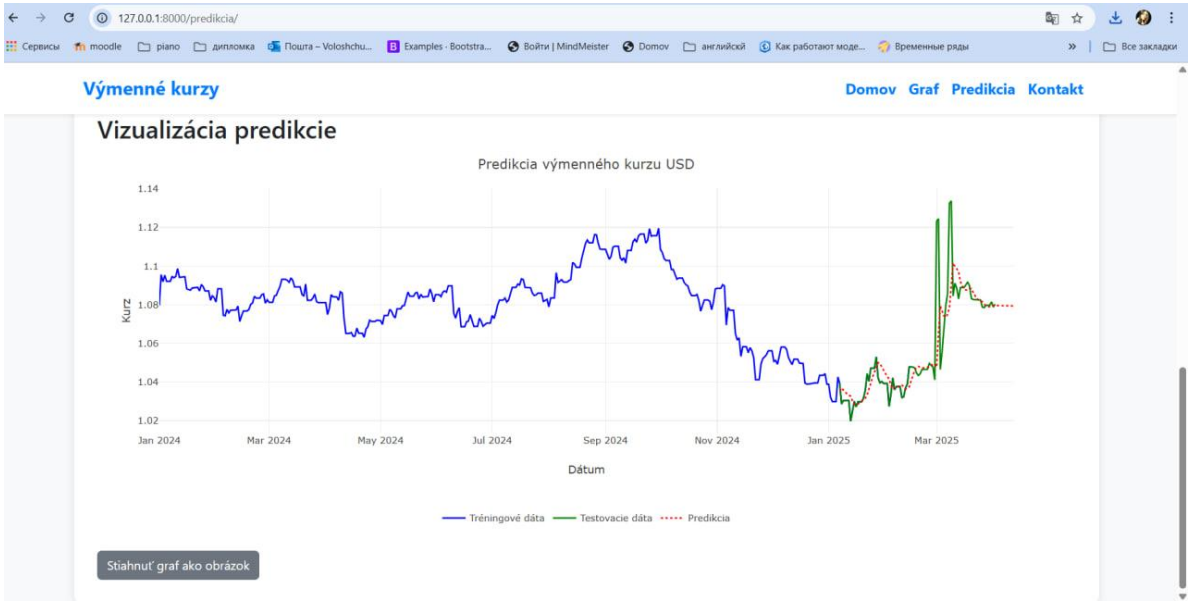
Po výbere modelov sme pristúpili k ich integrácii do webovej aplikácie vytvorenej v Django. Každý model je zapracovaný ako samostatná funkcia zodpovedná za načítanie historických údajov z databázy, výpočet predikcie na testovacej vzorke a výpočet budúceho vývoja výmenného kurzu. Cieľom bolo zabezpečiť, aby aplikácia umožňovala používateľom v reálnom čase zobrazovať predikcie kurzov priamo na webovej stránke.

Architektúra integrácie je rozdelená do niekoľkých komponentov:

- **predictions.py** obsahuje centrálnu funkciu save_predictions`(currency, model_name, days)`, ktorá zvolí vhodný model podľa názvu `(lstm, linear_regression, prophet, timegpt)` a zavolá príslušnú predikčnú funkciu. Výsledky sa ukladajú do databázy v tabuľke Prediction, čím sa predchádza potrebe výpočtu pri každom požiadavku.
- **views.py** implementuje pohľad predictions_view, ktorý reaguje na požiadavky používateľa zo stránky (napr. výber meny, modelu, počtu dní predikcie). Tento view volá save_predictions, načítava dáta z databázy a pripravuje ich na vizualizáciu pomocou JavaScriptového frontendového komponentu. Všetky modely musia vracať

testovacie metriky (MAE, MSE, RMSE, R2), predikciu a dátumy vo formáte kompatibilnom s grafom.

- **urls.py** obsahuje cesty, ktoré smerujú požiadavky používateľov na príslušné funkcie vo views.py.



Obrázok 57: Vizualizácia predikcie na lokálnom serveri
Zdroj: vlastné spracovanie

Týmto krokom bola ukončená **lokálna integrácia modelov** a pripravili sme aplikáciu na nasadenie do produkčného prostredia.

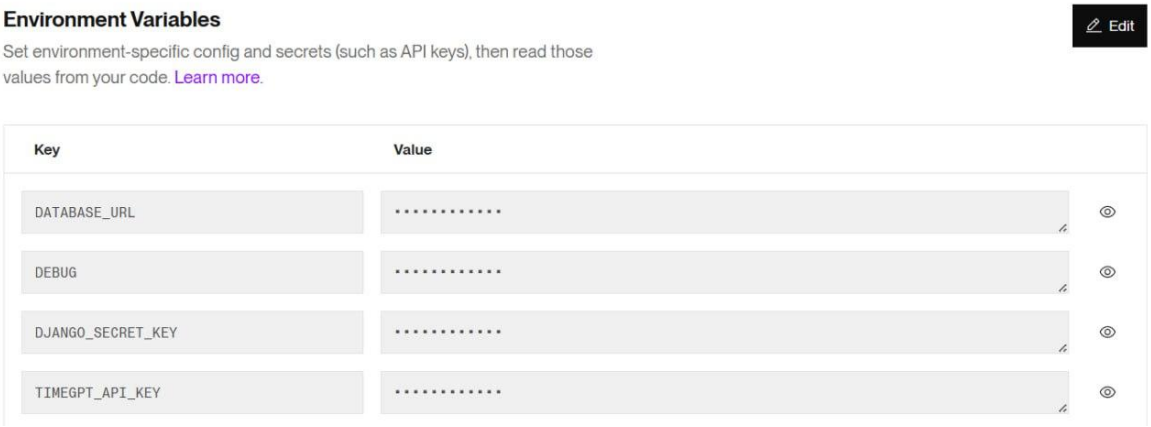
4.5 Nasadenie webovej aplikácie na Render

Pre nasadenie webovej aplikácie do produkčného prostredia bola zvolená cloudová platforma **Render**, ktorá umožňuje jednoduché a spoľahlivé nasadenie webových aplikácií priamo z repozitára verzovacieho systému GitHub. Táto platforma poskytuje výhodu v tom, že výrazne zjednodušuje proces deploymentu bez potreby manuálneho nastavovania servera, pričom automaticky spravuje spúšťanie aplikácie a jej škálovanie.

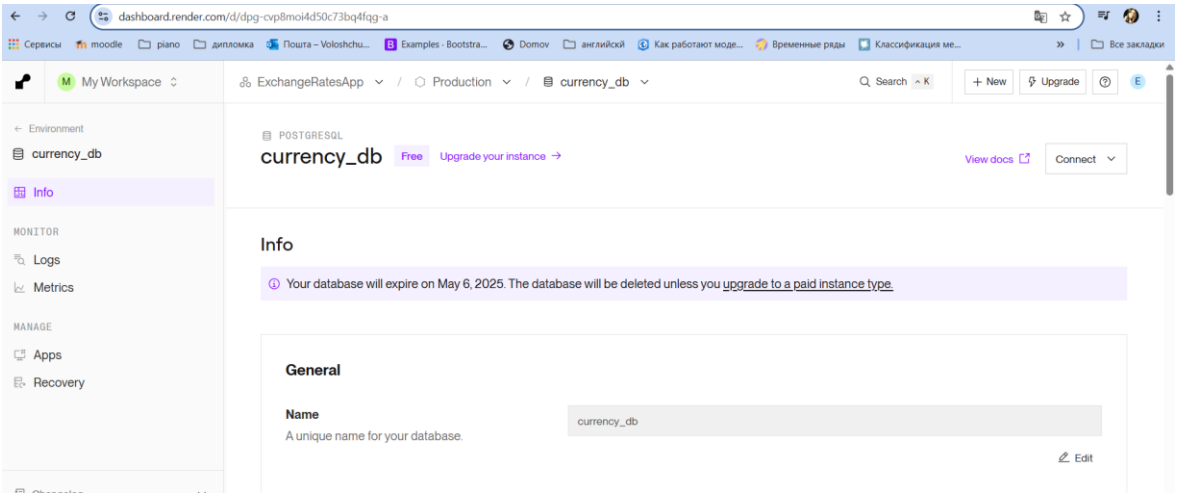
Proces nasadenia aplikácie pozostával z niekoľkých krokov. Najskôr bol projekt prepojený s platformou Render prostredníctvom GitHub repozitára https://github.com/vvvvElya/exchange_rates_app/tree/main .

Následne boli v Render definované základné parametre služby, vrátane Build Command a Start Command potrebných na správne zostavenie a spustenie aplikácie v prostredí kontajnera.

Render navyše poskytuje možnosť definovať premenné prostredia (Obr. 58), ktoré boli využité na bezpečné uloženie citlivých konfigurácií, ako sú prihlasovacie údaje do databázy a API kľúč pre model TimeGPT.



Obrázok 58: Nastavenie premenných prostredia v Render
Zdroj: vlastné spracovanie



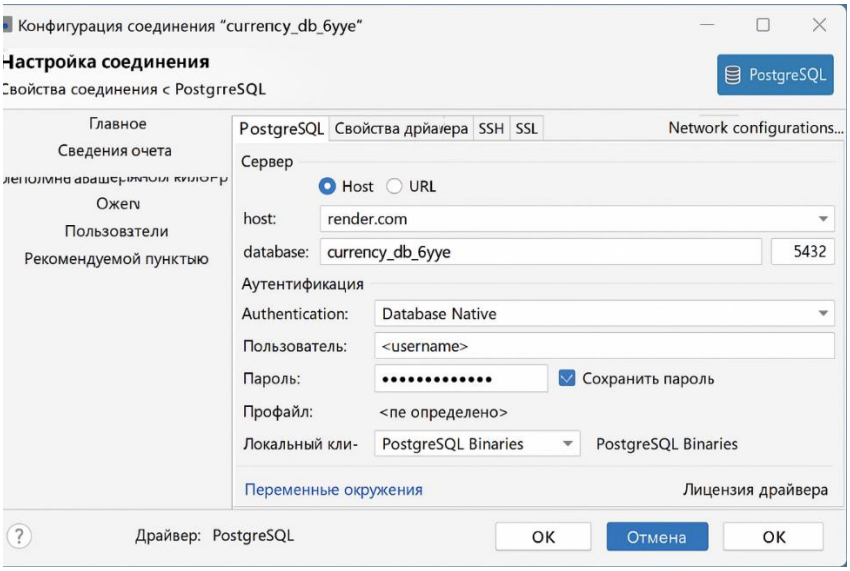
Obrázok 59: Vytvorenie databázy PostgreSQL na platforme Render
Zdroj: vlastné spracovanie

Pri vytváraní databázy bol zvolený názov currency_db, ktorý reprezentuje centrálnu databázu projektu pre predikciu výmenných kurzov (Obr. 59).

Na správu databázy PostgreSQL v rámci projektu bol použitý nástroj **DBeaver**. Tento open-source databázový klient umožňuje jednoduchú vizualizáciu štruktúry databázy, správu tabuliek, vykonávanie SQL dopytov a export alebo import údajov.

Na Obr. 60 je znázornené nastavenie pripojenia k produkčnej databáze PostgreSQL, ktorá bola nasadená v cloudovom prostredí Render. Konfigurácia zahŕňa definíciu databázového hostiteľa, názvu databázy, prihlasovacích údajov a portu, ktorý je pre PostgreSQL štandardne nastavený na 5432. Citlivé údaje boli pre účely tejto práce

anonymizované. Takáto konfigurácia umožňuje bezpečné a spoľahlivé pripojenie medzi aplikáciou a databázovým serverom, pričom prihlasovacie údaje sú uložené v environment variables, čím sa zvyšuje bezpečnosť systému.

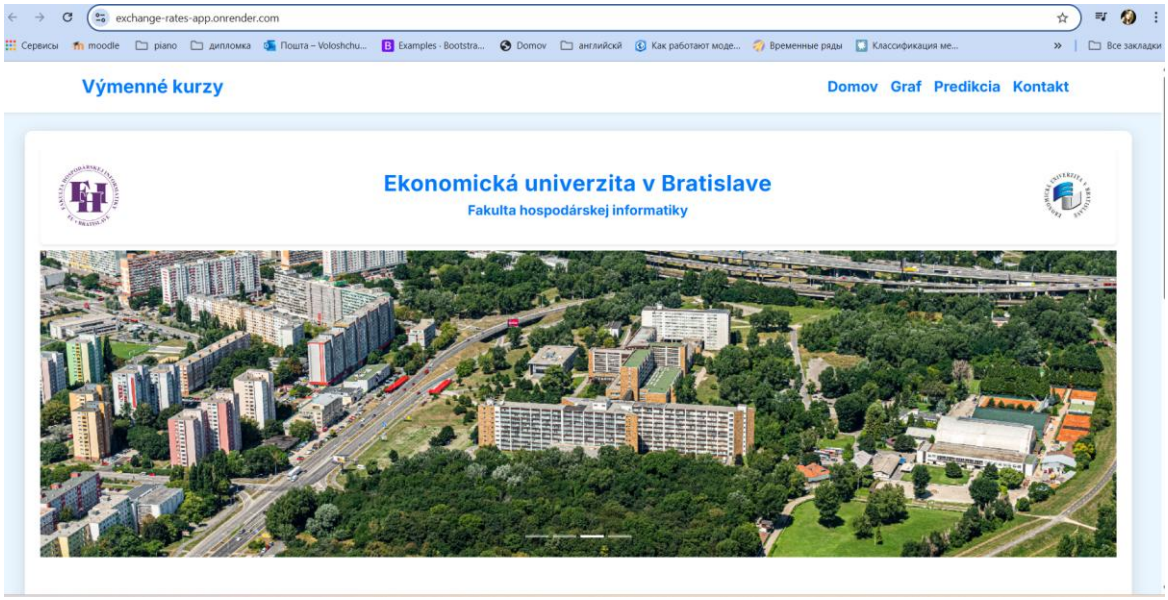


Obrázok 60: Konfigurácia pripojenia k databáze PostgreSQL
Zdroj: vlastné spracovanie

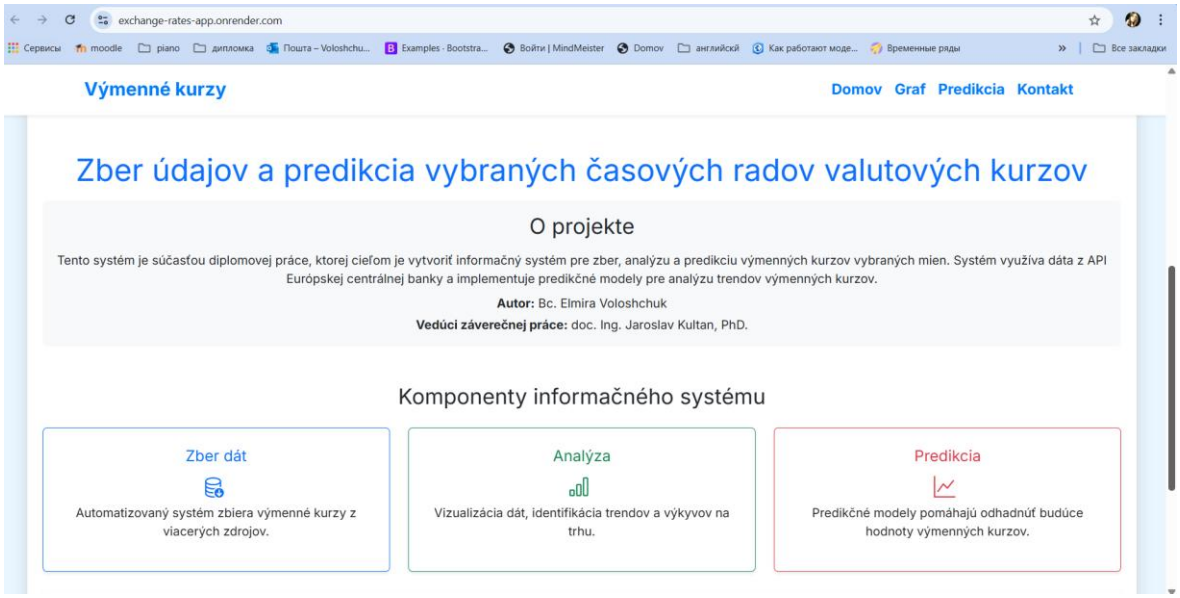
Výsledkom týchto krokov je funkčná webová aplikácia dostupná verejne prostredníctvom URL adresy pridelenou platformou Render, ktorá umožňuje používateľom zobrazovať predikcie výmenných kurzov priamo cez prehliadač.

https://exchange-rates-app.onrender.com

Po nasadení aplikácie do produkčného prostredia nasledovalo overenie funkčnosti a prezentácia používateľského rozhrania.

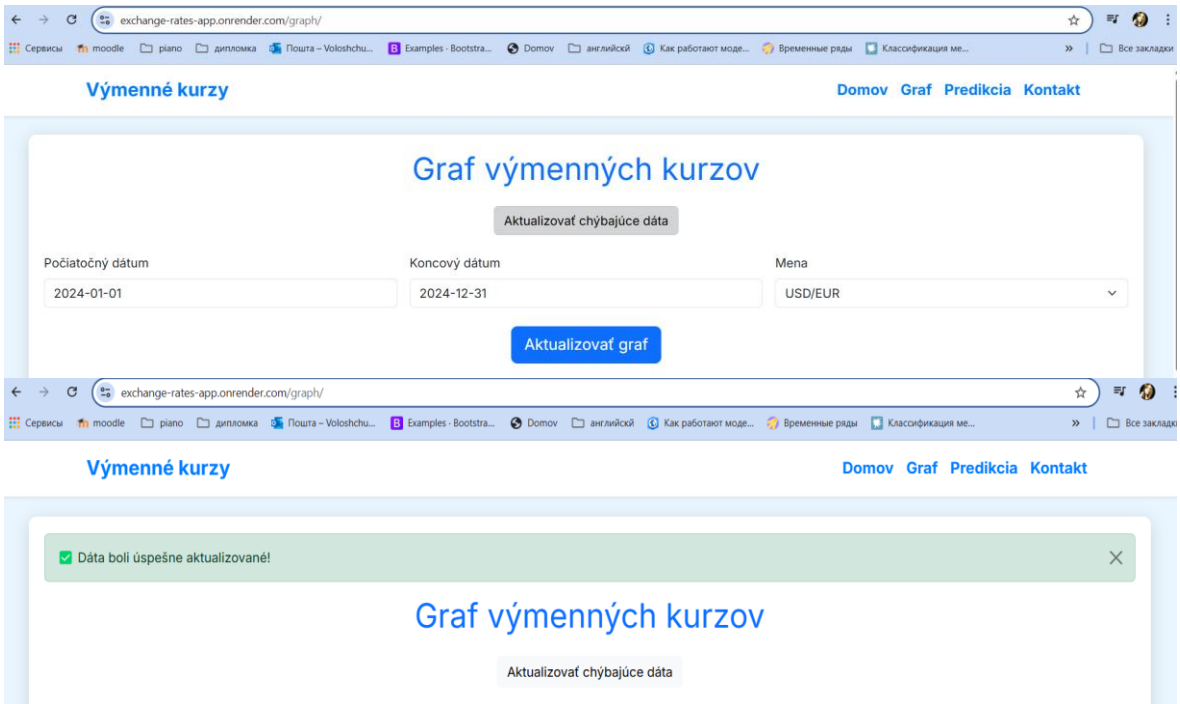


Obrázok 61: Úvodná obrazovka webovej aplikácie dostupnej na platforme Render
Zdroj: vlastné spracovanie – (URL: https://exchange-rates-app.onrender.com)



Obrázok 62: Úvodná obrazovka webovej aplikácie — o projekte
Zdroj: vlastné spracovanie – (URL: https://exchange-rates-app.onrender.com)

Vzhľadom na obmedzenia platformy Render a technické limity integrácie Celery v tomto prostredí, bola vytvorená samostatná funkcia **backfill_missing_data**. Táto funkcia umožňuje doplniť chýbajúce údaje o menových kurzoch prostredníctvom API Frankfurter. ktorý poskytuje menové kurzy priamo z ECB aj spätne za dni, pre ktoré predchádzajúce automatizované spracovanie zlyhalo alebo neprebehlo. Okrem historických dát zabezpečuje aj získanie a uloženie aktuálnych kurzov do databázy.



Obrázok 63: Aktualizácia chýbajúcich dát
Zdroj: vlastné spracovanie – (URL: https://exchange-rates-app.onrender.com)

Použitie TensorFlow Lite pre produkčné nasadenie modelu

Okrem toho pre účely produkčného nasadenia predikčných modelov bola zvolená optimalizovaná verzia TensorFlow Lite. Tento framework je navrhnutý špecificky pre efektívne spúšťanie modelov strojového učenia v prostredí s obmedzenými zdrojmi, ako sú cloudové služby alebo mobilné zariadenia. V rámci projektu bol model neurónovej siete LSTM konvertovaný do formátu TensorFlow Lite, čo umožnilo výrazné zníženie veľkosti modelu a zrýchlenie procesu predikcie. Tento krok bol kľúčový najmä vzhľadom na nasadenie aplikácie na platforme Render, kde je dôležité minimalizovať zaťaženie systému a zabezpečiť plynulé generovanie predikcií v reálnom čase.

Tabuľka 11: Porovnanie parametrov modelov TensorFlow
Zdroj: vlastné spracovanie

Porovnanie parametrov modelov	Pôvodný model (Keras H5)	Optimalizovaný model (TensorFlow Lite)
Formát súboru	.h5	.tflite
Veľkosť modelu (na menu)	~ 5–15 MB	~ 0.5–2 MB
Spotreba pamäte RAM (pri načítaní)	~ 30–50 MB	~ 2–5 MB
Rýchlosť načítania modelu	Stredná	Vysoká
Rýchlosť predikcie	Stredná	Vysoká
Vhodnosť pre produkčné prostredie	Obmedzená	Vysoká
Podpora na mobilných / edge zariadeniach	Nízka	Vysoká

Vizualizácia funkcionality predikcii

V časti „Predikcia výmenného kurzu“ má používateľ možnosť interaktívne pracovať s predikčným modulom aplikácie. Rozhranie umožňuje:

- výber meny,
- voľbu požadovaného modelu predikcie (napr. Prophet, LSTM, TimeGPT a LR),
- a zadanie počtu dní predikcie, pričom maximálny povolený horizont je 30 dní.

Po kliknutí na tlačidlo „Zobraziť predikciu“ sa zobrazí graf s vizualizáciou historických, testovacích aj predikovaných údajov. Aplikácia zároveň poskytuje:

- číselný výpis predikovaných hodnôt pre budúce obdobie,
- metriky presnosti na testovacej množine (napr. MAE, RMSE, MSE),

- a možnosť stiahnuť graf predikcie vo formáte PNG, čím sa zvyšuje praktická využiteľnosť nástroja v profesionálnom aj akademickom prostredí.

Obrázok 64:Graf predikcie kurzu USD cez webové rozhranie – model Prophet
Zdroj: vlastné spracovanie – (URL: https://exchange-rates-app.onrender.com)

Obrázok 65: Graf predikcie kurzu GBP cez webové rozhranie – model LR
Zdroj: vlastné spracovanie – (URL: https://exchange-rates-app.onrender.com)

Záver

Diplomová práca bola zameraná na návrh a implementáciu informačného systému na zber, spracovanie a predikciu časových radov valutových kurzov. Hlavným cieľom bolo vytvoriť funkčné riešenie, ktoré umožní zber aktuálnych kurzových údajov, ich ukladanie do databázy a následné využitie pre predikciu budúcich hodnôt.

V rámci práce bol navrhnutý a implementovaný systém založený na webovom frameworku Django s databázou PostgreSQL. Zber údajov bol zabezpečený prostredníctvom integrácie s API Európskej centrálnej banky, ako aj alternatívnym spôsobom pomocou VBA makier v Exceli a Plánovača úloh systému Windows. Súčasťou riešenia bolo aj zabezpečenie kontroly nad priebežnou aktualizáciou údajov a doplnením chýbajúcich záznamov v databáze. Na predikciu boli otestované viaceré modely – od jednoduchšej lineárnej regresie a modelu Prophet až po pokročilé riešenia ako LSTM neurónové siete a TimeGPT. Systém bol nasadený na platformu Render, čo umožnilo zabezpečiť dostupnosť webovej aplikácie a vizualizáciu výsledkov predikcií v reálnom čase vo forme prehľadných a interaktívnych grafov. Zdrojový kód riešenia je verejne dostupný na platforme GitHub: **https://github.com/vvvvElya/exchange_rates_app/tree/main** , zatiaľ čo samotná webová stránka aplikácie je prístupná online na adrese: https://exchange-rates-app.onrender.com/

Práca zároveň identifikovala niekoľko možností pre budúce rozšírenie systému, medzi ktoré patrí napríklad zavedenie ďalších predikčných modelov, podpora multivariantných časových radov, ktoré by pri predikcii zohľadňovali nielen historické údaje o samotnom menovom kurze, ale aj ďalšie relevantné faktory, ako sú úrokové sadzby, inflácia či geopolitické ukazovatele. Doplnkovo by bolo možné využiť aj analýzu sentimentu z finančných správ alebo sociálnych médií, čo by mohlo zvýšiť citlivosť predikcií na aktuálne udalosti. Navyše, navrhnutá architektúra systému je dostatočne flexibilná na to, aby umožnila integráciu ďalších dátových zdrojov alebo nasadenie v produkčnom prostredí s vyššou záťažou.

Prínosom práce je vytvorenie funkčného systému, ktorý prepája automatizovaný zber dát, ich spracovanie a predikciu s prehľadnou vizualizáciou výsledkov. Používateľ tak získa rýchly prístup k spoľahlivým historickým aj budúcim údajom o vývoji kurzov, čo podporuje kvalitnejšie rozhodovanie a riadenie menového rizika. Práca poskytuje komplexný pohľad na celý proces spracovania kurzových údajov — od ich získavania v reálnom čase až po prezentáciu výsledkov prostredníctvom webového rozhrania.

Zoznam použitej literatúry

AL-SHBOUL, M. *A Comparative Study of ARIMA and Machine Learning Models for Currency Exchange Rate Prediction.*. Journal of Advanced Computer Science, 2020.

AMAZON WEB SERVICES. (2024). What is streaming data? [online]. Dostupné na: https://aws.amazon.com/what-is/streaming-data/

APACHE FOUNDATION. *Apache Kafka Documentation*. Dostupné na: apache

AWAN, A. A. (2024, September 2). *Time series forecasting with TimeGPT. DataCamp.* https://www.datacamp.com/tutorial/time-series-forecasting-with-time-gpt

BOLLERSLEV, T. *Generalized Autoregressive Conditional Heteroskedasticity.* Journal of Econometrics, 1986.

BOOTSTRAP. (2023). Bootstrap Documentation [online]. Dostupné na: https://getbootstrap.com/docs/4.1/getting-started/introduction

BOX, G. E. P. – JENKINS, G. M. *Time Series Analysis: Forecasting and Control.* Holden-Day, 1976.

BULLEN, S. – BOVEY, R; GREEN, J. *Professional Excel Development: The Definitive Guide to Developing Applications Using Microsoft Excel and VBA*. Boston: Addison-Wesley Professional, 2005. ISBN 9780321262509.

CELERY. *Dokumentácia k knižnici Celery* [online]. Dostupné na: https://docs.celeryproject.org

CHATFIELD, C. *The Analysis of Time Series: An Introduction. Chapman and Hall/CRC,* 2004

CHEN, et al. *Transformer-based Models for Time Series Forecasting.* Journal of Machine Learning Research, 2024.

ČUČUEVA, I.A. *Model prognozirovania vremenných riadov po vyborke maksimalnogo podobija*. Moskva, 2012.

DEGTIAREV, K. *Prognózovanie výmenného kurzu USD/RUB pomocou fuzzy časových radov*, 2006.

DJANGO SOFTWARE FOUNDATION. (2023). Django REST framework Documentation. Dostupné na: https://www.django-rest-framework.org

DURBIN R. et al., *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

ENGLE, R. F. *Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation*. Econometrica, 1982.

ESTELLÉS-AROLAS, E., & GONZÁLEZ-LADRÓN-DE-GUEVARA, F. (2012). *Towards an integrated crowdsourcing definition. Journal of Information Science,* 38(2), 189–200.

EUROPEAN CENTRAL BANK. (2024). *Euro foreign exchange reference rates* [online]. Dostupné na: Euro foreign exchange reference rates

EUROPEAN COMMISSION REPRESENTATION IN SLOVAKIA. (2023, October 17). *Digitalizácia v Európe. European Commission* [online]. Dostupné na: https://slovakia.representation.ec.europa.eu/news/digitalizacia-v-europe-2023-10-17_sk

FIELDING, R. *Architectural Styles and the Design of Network-based Software Architectures.* University of California, Irvine, 2000.

FLATLOGIC. Best API Integration Practices for Modern Web Applications. Flatlogic Blog [online]. 2024. [cit. 2025-04-14]. Dostupné na: https://flatlogic.com

GARZA, Z., LIU, X., & WANG, J. (2023). Temporal fusion transformers for interpretable time series forecasting: A review. arXiv preprint arXiv:2310.03589. https://arxiv.org/pdf/2310.03589

GEEKSFORGEEKS. Web Scraping Techniques and Tools. GeeksforGeeks [online]. 2023. Dostupné na: https://www.geeksforgeeks.org/introduction-to-web-scraping/?ref=shm#techniques-of-web-scraping

GELMAN, A. et al. – *Bayesian Data Analysis.* Chapman and Hall/CRC, 2014.

HAMILTON, J. D. *Time Series Analysis.* Princeton University Press, 1994.

HOCHREITER, S., & SCHMIDHUBER, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. Dostupné na: https://doi.org/10.1162/neco.1997.9.8.1735

HUDEC, M. *Fuzziness in Information Systems: How to Deal with Crisp and Fuzzy Data in Selection, Classification, and Summarization*. Cham: Springer, 2016. ISBN 978-3-319-42516-0. Dostupné na: https://doi.org/10.1007/978-3-319-42518-4

HWANG, R.-J., **CHEN**, S.-M., **LEE**, C.-H. *Handling forecasting problems using fuzzy time series*. Fuzzy Sets and Systems, 1998, roč. 100, s. 217–228. ISSN 0165-0114. Dostupné na: https://doi.org/10.1016/S0165-0114(97)00126-0

HYNDMAN, R. J. – ATHANASOPOULOS, G. *Forecasting: Principles and Practice.* OTexts [online]. 2018. [cit. 2025-04-14]. Dostupné na: https://otexts.com/fpp3/

INTERNATIONAL DATA CORPORATION (IDC). (2024). *IDC FutureScape: Worldwide AI and Automation 2024 Predictions* [online]. Dostupné na: https://my.idc.com/getdoc.jsp?containerId=IDC_P33195

ITAIMS (2023). What is Internet of Things? [online]. Dostupné na: iot itams

JAMES, G. – WITTEN, D. – HASTIE, T. – TIBSHIRANI, R. *An Introduction to Statistical Learning.* Springer, 2021.

JANSSEN, K. *Open Data Portals and Their Impact on Open Government.* Government Information Quarterly, 2012.

KIPF, T. N., & WELLING, M. (2017). *Semi-supervised classification with graph convolutional networks.* In International Conference on Learning Representations (ICLR).

KOENKER, R. *Quantile Regression.* Cambridge University Press, 2005.

KUMAR, V., & DEEPAK, S. (2024). *Leveraging robotic process automation (RPA) for end-to-end testing in Agile and DevOps environments:* A comparative study. Online Scientific Research [online]. Dostupné na: RPA

KUNE, R., KONUGURTHI, P. K., AGARWAL, A., CHILLARIGE, R. R., & BUYYA, R. (2016). *The anatomy of Big Data computing. Software: Practice and Experience,* 46(1), 79–105. https://doi.org/10.1002/spe.2374

MANUKONDA, K. (2024). *Leveraging robotic process automation (RPA) for end-to-end testing in Agile and DevOps environments: A comparative study*. Journal of Artificial Intelligence & Cloud Computing, https://doi.org/10.47363/JAICC/2024(3)315

MEDJAOUI, M. – WILDE, A. – MITRA, R. – AMUNDSEN, M. *Continuous API Management.* O'Reilly Media, 2018.

MICROSOFT. *Dokumentácia k Power Query* [online]. 2025. Dostupné na: https://learn.microsoft.com/sk-sk/power-query/

MITCHELL, R. *Web Scraping with Python: Collecting More Data from the Modern Web.* O'Reilly Media, 2018.

MITRANI, S. *Achieving stationarity with time series data*. Medium [online]. 2023. Dostupné na: data science- medium

MONTGOMERY, D. C. – PECK, E. A. – VINING, G. G. *Introduction to Linear Regression Analysis.* Wiley, 2012.

NUMPY. NumPy Documentation [online]. Dostupné na: https://numpy.org

OLAH, C. *Understanding LSTM Networks.* Colah's Blog [online]. 2015. [cit. 2025-04-14]. Dostupné na: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

PALLETT, M. (2022). Flask Web Development. O'Reilly Media.

PANDAS *Pandas Documentation*. 2023. Dostupné na: https://pandas.pydata.org

PLOTLY Creating interactive charts with Plotly.js. Dostupné na: https://plotly.com

PROPHET. *Prophet Documentation*. 2023 Dostupné na: https://facebook.github.io/prophet

QURESHI, I. *API Architecture* . Medium, 2023. Dostupné na: APIs play a fundamental role in cloud services. | by Majid Qureshi | Medium

QUIX. *Data Streaming in Real-Time Applications.* Quix Blog [online]. 2021. Dostupné na: https://quix.io

ROSAM. *Streaming Data Architecture.* Tech Sparks [online]. 2021. Dostupné na: https://techsparks.com

ROSS, S. M. (1995*). Stochastic Processes.* 2nd ed. John Wiley & Sons.

RUBLIKOVÁ, E. *Analýza časových radov*. IURA Edition, Bratislava , s. 207., 2007. ISBN 978-80-8078-139-2.

RUBLIKOVÁ, E. – PRIHODOVÁ, J. *Analýza časových radov vo finančnej praxi.* IURA Edition, 2008.

RUBLIKOVÁ, E. – PRIHODOVÁ, J . *Analýza vybraných časových radov - ARIMA modely.* Vydavateľstvo EKONÓM, 2008.

SCIKIT-LEARN (2023). *Scikit-learn Documentation* [online]. Dostupné na: scikit-l

SIVABALAN, S. *ETL vs ELT: What's the Difference?* Data Engineering Blog [online]. 2021. Dostupné na: https://dataengineering.com

SONG, Q., **CHISSOM**, B. S. *Forecasting enrollments with fuzzy time series – Part I*. Fuzzy Sets and Systems, 1993. Dostupné na: https://doi.org/10.1016/0165-0114(93)90355-L

STATSMODELS. *Statsmodels Documentation* [online]. Dostupné na: statsmodels

STOCK, J. H. – WATSON, M. W. *Vector Autoregressions.* Journal of Economic Perspectives, 2001.

TANG, Y. *Deep Learning for Time Series Forecasting.* Springer, 2023.

TENSORFLOW (2024). *TensorFlow/Keras Documentation* [online]. Dostupné na: https://www.tensorflow.org

VAN VEEN, F. – LEIJNEN, S. *Neural Network Architectures Illustrated.* Machine Learning Mindset [online]. 2019. Dostupné na: https://www.asimovinstitute.org/neural-network-zoo

WANG, H. *Advances in Neural Networks for Time Series Analysis.* Artificial Intelligence Review, 2024.

WHITENACK, D. – SELVARAJ, V. *Time Series Forecasting: Principles and Practices.* MLJ Books, 2019.

W3SCHOOLS. *HTML and CSS Guide.*. Dostupné na: https://www.w3schools.com

ZADEH, L. A. *Fuzzy sets*. Information and Control, 1965, roč. 8, č. 3, s. 338–353. ISSN 0019-9958. Dostupné na: https://doi.org/10.1016/S0019-9958(65)90241-X

ZULKIFLI, A. *Data Integration with SQL Server Integration Services.* Data Management Journal, 2021.