VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PŘEDPOVÍDÁNÍ VÝVOJE ČASOVÝCH ŘAD

TIME SERIES PREDICTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ DVOŘÁČEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. DAVID HŘÍBEK,**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

148577

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Dvořáček Tomáš**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Předpovídání vývoje časových řad**
Kategorie:	Umělá inteligence
Akademický rok:	2022/23

Zadání:

1. Nastudujte problematiku předpovídání časových řad, kdy se ze sady dat předpovídá vývoj jedné, případně i více proměnných.
2. Navrhněte program, který načte data časových řad a pro vybrané řady bude předpovídat jejich budoucí hodnoty ve zvolené časové vzdálenosti.
3. Navržený program implementujte.
4. Testování proveďte na sadě dat, kde budou jednoduchá data jako např. funkce sinus, tak i složitější, vícedimenzionální data, kde se budou předpovídat jen vybrané řady.

Literatura:

- Bryan Lim and Stefan Zohren: Time-series forecasting with deep learning: a survey, Transactions of the Royal Society A, April, 2021, Online ISSN:1471-2962
- Christopher M. Bishop, ISBN: 1493938436, Pattern Recognition and Machine Learning, 2016.

Při obhajobě semestrální části projektu je požadováno:
První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Hříbek David, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Cílem této práce je navrhnout a implementovat program, který ze zadaného vstupu bude schopen analyzovat a predikovat budoucí vývoj univarietních a multivarietních časových řad. V řešení byly využity statistické přístupy a přístupy kdy se časová řada předpovídá pomocí neuronových sítí.

Abstract

The aim of this thesis is to design and implement a program that will be able to analyze and predict the future evolution of univariate and multivariate time series from a given input. Statistical approaches and approaches where time series are predicted using neural networks have been used in the solution.

Klíčová slova

časové řady, předpovídání, neuronové sítě, rekurentní neuronové sítě, konvoluční neuronové sítě, normalizace, rozhraní, stacionarita, statistické modely, Box-Jenkins, ARIMA, SARIMA, VAR

Keywords

time series, forecasting, neural networks, recurent neural networks, convolutional neural networks, interface, stationarity, statistical models, Box-Jenkins, ARIMA, SARIMA, VAR

Citace

DVOŘÁČEK, Tomáš. *Předpovídání vývoje časových řad.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. David Hříbek,

Předpovídání vývoje časových řad

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Davida Hříbka Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Dvořáček
7. května 2023

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Davidu Hříbkovi za pomoc a dobré rady při vedení mé bakalářské práce.

Obsah

Kapitola 1

Úvod

Časové řady se používají pro analýzy nebo predikce v mnoha různých odvětvích jako je finančnictví, sociální vědy, zdravotnictví či meteorologie a klimatologie. V podstatě se dá říct že predikci pomocí časové řady můžeme aplikovat v jakémkoli odvětví, které zahrnuje časová měření. Analyzování časových řad má značný komerční potenciál, vzhledem k tomu že se dá využít například pro částečnou predikci vývoje na akciových trzích nebo pro predikce budoucí poptávky po určitém zboží a tím pádem jeho prodejů. Rostoucí důležitost tvorby těchto prognóz nejenom v komerčním oboru může potvrzovat i fakt, že systémy řízení báze dat založené na časových řadách, jsou jednou z nejrychleji rostucích oblastí v databázovém průmyslu, za poslední dva roky tato oblast dokonce zaznamenala vůbec nejrychlejší růst.

Analýzou časové řady můžeme získat mnoho cenných statistických údajů a charakteristik, jež můžou pomoci k lepšímu porozumění zkoumaného souboru dat. Analýzou myslíme tvorbu postupů, jež povedou k lepšímu pochopení podstaty chování sledovaných dat a tvorbu modelů použitelných pro předpovědi budoucích hodnot v čase, jež jsou založeny na zkoumaných datech. Samotná implementace modelů a vytvoření programu, který bude předpovídat budoucí hodnoty je smyslem a účelem této práce.

Kapitola 2

Časová řada

Časová řada je soubor datových bodů, měřených obvykle v po sobě jdoucích časových úsecích. Matematicky ji lze zadefinovat jako množinu vektorů $\vec{x_t}, t = 0, 1, 2, ...$ kde t představuje uplynulý čas. S vektorem $\vec{x_t}$ se zachází jako s náhodnou proměnnou. Měření provedená během události v časové řadě jsou uspořádána ve správném chronologickém pořadí.

Časová řada může být spojitá nebo diskrétní. Ve spojité časové řadě jsou pozorování měřena v každém časovém okamžiku, zatímco diskrétní časová řada obsahuje pozorování měřená v diskrétních časových okamžicích. Příkladem spojité řady může být sběr meteorologických dat, u diskrétní řady třeba uzavírající cena svíčky na burzovním svíčkovém grafu. Spojitou časovou řadu lze transformovat na diskrétní, a to agregací dat v určitém časovém intervalu [7].

2.1 Klasifikace časových řad

Existuje mnoho různých druhů časových řad, pro představu uveďme alespoň krátký výčet:

- Pravidelné/Nepravidelné - Pozorovaná data mohou být shromažďována buď v pravidelných nebo nepravidelných časových intervalech, což má za následek vznik pravidelné či nepravidelné časové řady.
- Univarietní/Multivarietní - Univarietní a multivarietní časovou řadou rozumíme řadu, v níž se v čase mění jedna, respektive dvě a více časově závislých proměnných.
- Lineární/Nelineární - Lineární časová řada je taková, kde každou její hodnotu X_t lze považovat za lineární kombinaci minulých nebo budoucích hodnot.
- Deterministické/Nedeterministické - Deterministická řada nemá pravděpodobnostní ani náhodné aspekty. Vzhledem k tomu, že ji lze přesně popsat matematickým výrazem, je možné její budoucí hodnoty poměrně snadno predikovat. Drtivá většina zkoumaných řad však vykazuje spíše nedeterministické chování.

Dále můžeme časové řady rozdělit na stacionární a nestacionární. Hlavní rozdíl mezi těmito dvěmi je ten, že statistické vlastnosti (rozptyl, autokorelace, ...) u stacionární řady se v čase nemění, kdežto u nestacionární se mění. Protože z nestacionární řady můžeme získat výsledky, které mohou být falešné, např. mohou naznačovat vztah mezi dvěma proměnnými, který ve skutečnosti neexistuje, musíme takovou řadu pomocí různých metod transformovat na stacionární.

2.2 Skladba časové řady

Trendová složka

Trendová složka ukazuje, zda mají zaznamenané hodnoty dat v čase obecnou tendenci růst nebo klesat. Trend může být i stabilní, pak hovoříme o bočním trendu. Tendence zaznamenaných hodnot se může v různých časových úsecích značně odlišovat od svého průměru, v dlouhodobém horizontu však musí buď růst, klesat či být stabilní, aby se dala považovat za trend.

Sezonní složka

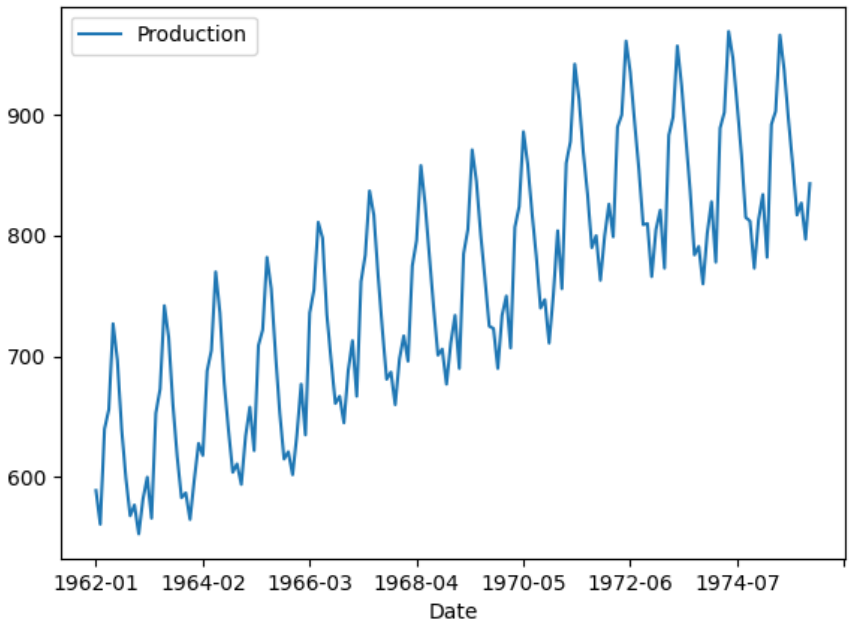
Sezonní složkou jsou myšleny pravidelné, periodické pohyby, které se opakují ve stejných časových intervalech. Tyto odchylky můžou vznikat např. jako důsledek přírodních sil nebo zvyků vytvořených lidstvem.

Cyklická složka

Cyklickou složku lze popsat stejně jako sezonní složku s tím rozdílem, že periodické vzorce v datech jsou nepravidelné a opakují se v náhodných časových intervalech. Z tohoto důvodu nelze tyto pohyby v časové řadě predikovat. Jedním z nejznámějších příkladů této složky jsou ekonomické cykly.

Náhodná složka

Prakticky každá časová řada má náhodnou složku, která se skládá z krátkodobých, náhodných a nesystematických výkyvů. Náhodná složka, též známá pod pod pojmem reziduum, je tím, co zůstane po extrakci všech výše popsaných složek z časové řady.



Obrázek 2.1: Příklad časové řady, která obsahuje trendovou a sezonní složku.

2.3 Stacionarita

Stacionární časovou řadou je myšlena řada, jejíž statistické vlastnosti jako střední hodnota či rozptyl jsou v čase konstantní. Většina předpovědních modelů je založena na předpokladu, že časovou řadu je možné pomocí různých matematických transformací "stacionarizovat". Předpovědi pro stacionarizované řady pak lze zpětně transformovat, a to obrácením všech dříve použitých matematických transformací, a získat tak předpověd pro původní řadu.

Zjištění posloupnosti transformací potřebných ke stacionarizaci časové řady je důležité při hledání vhodného prognostického modelu, např. stacionarizace časové řady pomocí diferenciace je důležitou součástí procesu pro přizpůsobení předpovědního modelu ARIMA. Samotná stacionarita může být dvojího typu:

1. Silná stacionarita
2. Slabá stacionarita

U silné stacionarity platí, že rozdělení pravděpodobnosti $(x_{t-n}, x_{t-n+1}, x_t, x_{t+n-1}, x_{t+n})$ je nezávislé na t pro všechna n. Pro praktické využití však není předpoklad silné stacionarity vždy nutný, navíc se poměrně těžko ověřuje. V praxi mnohem častěji stačí aby časová řada vykazovala slabou stacionaritu. U takové řady platí, že její střední hodnota a kovariance je nezávislá na čase, tedy $\mu_t = \mu$ a $Cov(x_t, x_{t-n})$ závisí pouze na n [11].

Bílý šum

Jedním z nejtypičtějších příkladů stacionární časové řady je bílý šum. Jeho hodnoty jsou zcela náhodné, neexistuje mezi nimi žádná korelace a rozložení hodnot odpovídá normálnímu rozdělení se středem v nule.

Náhodná procházka

Náhodná procházka je příklad nestacionární časové řady. Její hodnoty jsou generovány jako kumulativní součet předchozích hodnot řady, která odpovídá bílému šumu. Kovariance náhodné procházky roste do nekonečna, platí tedy že $\gamma_n = Cov(x_t, x_{t+n}) = t\sigma^2 \to \infty$ pro $n \to \infty$ [9].

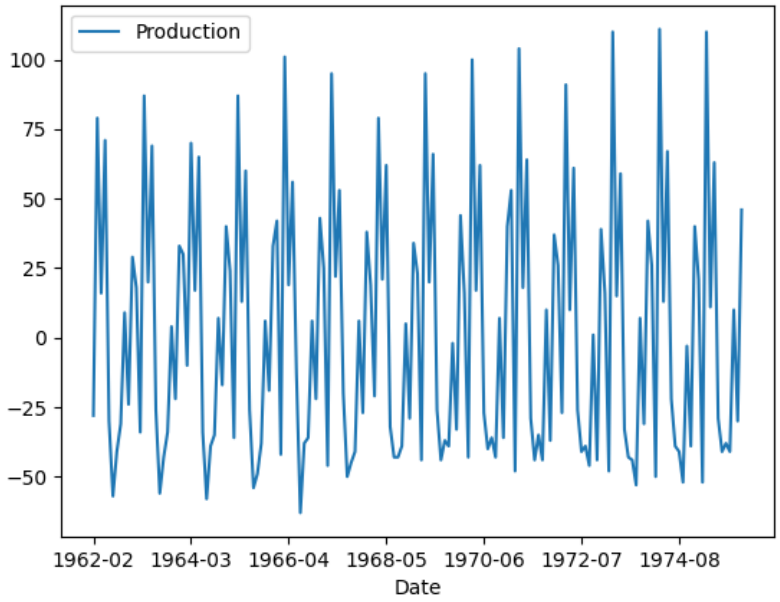
Rozšířený Dickeyho-Fullerův test

Rozšířený Dickeyho-Fullerův test (ADF) slouží k otestování stacionarity časové řady. Jedná se o statistický test významnosti, který poskytuje výsledky v testech hypotéz s nulovou a alternativní hypotézou. Za nulovou hypotézu je bráno tvrzení, že zkoumaná časová není stacionární. Její přijmutí nebo zamítnutí je dosaženo porovnáním p-hodnoty s kritickou hodnotou (typicky 0.05) a vypadá následovně:

1. p-hodnota > kritická hodnota: Nelze zamítnout nulovou hypotézu což znamená že časová řada není stacionární.
2. p-hodnota <= kritická hodnota: Nulová hypotéza se zamítá a je přijmuta alternativní hypotéza říkající že časová řada je stacionární.

Jednotkový kořen

Rozšířený Dickeyho-Fullerův test se řadí do kategorie tzv. testů jednotkového kořene. Podstatou testu je hledání přítomnosti jednotkového kořene ve zkoumané časové řadě. Jestliže je kořen nalezen, pak je časová řada nestacionární. Počet nalezených jednotkových kořenů odpovídá počtu diferenciací, které je potřeba provést aby se časová řada stala stacionární [16].



Obrázek 2.2: Časová řada z obrázku 2.1, jež byla transformována na stacionární.

2.4 Předpovídání vývoje statistickým přístupem

Předpovídáním vývoje časové řady je myšleno odhadování jejích budoucích hodnot, tedy pro řadu $x_1, x_2, ..., x_t$ předpovídáme koeficient x_{t+n}. Pro předpovídání vývoje časových řad existuje mnoho modelů a postupů, žádný z nich ale nelze univerzálně aplikovat na všechny časové řady. Aby bylo možné nějaký předpovědní model použít, je potřeba nejprve pochopit samotnou podstatu dat, ze kterých je časová řada složena.

Obecný přístup pro analýzu časových řad vypadá následovně [2]:

1. Vykreslení časové řady do grafu tak, aby bylo co nejlépe vidět jaké složky řada obsahuje nebo jestli jsou v datech ostré výkyvy.

2. Odstranění případné trendové či sezonní složky, a to pomocí různých matematických transformací nebo použitím diferenciace. Účelem je získat stacionární residuální složku.

3. Výběr vhodného předpovědního modelu a jeho aplikace na residuální složku, získanou v předchozím kroku.

4. Na získaných předpovězených hodnotách provést zpětnou transformaci.

Postup předpovídání se také liší od toho, jestli je zkoumaná řada univarietní nebo multivarietní. V prvním případě je predikce vývoje řady založena pouze na minulých hodnotách jedné řady, ve druhém případě je predikce závislá na hodnotách z více časových řad. Tyto hodnoty se pak označují jako tzv. prediktory. Při tvorbě adekvátního předpovědního modelu je nutné brát v potaz jeho složitost. Dobrý model by měl být co nejjednodušší a zároveň by měl být schopen co nejpřesněji modelovat chování časové řady. Komplexní modely dokážou velmi přesně modelovat trénovací data, nicméně pro predikce budoucích hodnot mohou vykazovat takové odchylky, že jsou v podstatě nepoužitelné [6]. Jedná se tedy o klasický problém overfittingu.

2.4.1 Statistické předpovědní modely

Autoregresní model AR(p)

Autoregresní model se při předpovídání aktuálních hodnot opírá pouze o hodnoty z minulých období. Jedná se o lineární model, kde aktuální hodnoty odpovídají součtu minulých hodnot vynásobených určitým číselným koeficientem. Model se označuje jako AR(p), kde "p"je řád modelu a představuje počet zpožděných hodnot, které chceme zahrnout. Parametr p lze vhodně zvolit pomocí grafu parciální autokorelace (PACF). Autoregresní model se zapisuje rovnicí

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + w_t \tag{2.1}$$

kde $\phi_{1,...,p}$ jsou paramentry modelu, X_t jsou hodnoty řady a w_t odpovídá normálně rozdělenému bílému šumu se středem v nule.

Model klouzavého průměru MA(q)

Model klouzavého průměru je model časové řady, který říká, že příští hodnota je průměrem všech předcházejících hodnot. Řád modelu klouzavého průměru, typicky označovaný jako q, lze obvykle odhadnout pohledem na autokorelační graf časové řady (ACF). U tohoto modelu jsou hodnoty časové řady X_t lineární funkcí všech současných a předchozích hodnot šumových veličin $w_1, ..., w_{t-1}$. Matematicky vypadá model takto

$$X_t = \mu + w_t + \theta_1 w_1 + \theta_2 w_2 + ... + \theta_{t-q} w_{t-q} \tag{2.2}$$

kde μ odpovídá střední hodnotě časové řady, $\theta_{1,...,q}$ jsou parametry modelu a w_t jsou, podobně jako v předchozím modelu, hodnoty normálně rozděleného bílého šumu.

Autoregresní integrovaný model s klouzavým průměrem ARIMA(p,d,q)

Jak může napovídat již samotný název, model ARIMA vznikl složením modelů AR(p) a MA(q). Bohužel, oba tyto modely jsou aplikovatelné jen na stacionární časové řady[2]. Problém se stacionaritou u modelu ARIMA řeší právě část "I"s parametrem d, který určuje řád diferenciace, tedy kolikrát byly hodnoty dat nahrazeny rozdílem mezi jejich aktuálními hodnotami a předchozími hodnotami. ARIMA je generalizovaný model, například ARIMA(p,0,0) odpovídá modelu AR(p).

Sezónní autoregresní model s klouzavým průměrem SARIMA(p,d,q)(P,D,Q,m)

SARIMA je kombinací dvou již existujících modelů, a to modelů ARIMA a sezonního autoregresivního modelu (SAR). SARIMA je schopna identifikovat a predikovat sezónní trendy v časových řadách a její vnitřní fungování zahrnuje tři základní komponenty: autoregresní komponentu (AR), integrační komponentu (I) a sezonní autoregresní komponentu (SAR). Autoregresní komponenta se soustředí na detekci souvislostí mezi hodnotami v časové řadě. Integrovaná komponenta se pak zaměřuje na diferencování dat, aby se odstranila sezónní složka. Nastavení hyperparametrů sezonní autoregresní komponenty (P,D,Q,m) pak ovlivňuje modelování sezónních trendů. SARIMA model je velmi flexibilní a může být použit k predikci budoucích hodnot časových řad s různými frekvencemi, jako je například denní, týdenní nebo měsíční.

Model vektorové autoregrese VAR(p)

Model vektorové autoregrese je jedním z nejrozšířenějších a nejsnáze použitelných modelů pro analýzu vícerozměrných časových řad. Je přirozeným zobecněním modelu AR(p) na dynamické vícerozměrné časové řady. VAR model pro n-proměnných se bude skládat ze soustavy n-rovnic. Každá rovnice bude obsahovat p zpožděných hodnot, podle řádu modelu. VAR(p) je vzhledem ke schopnosti modelovat více proměnných užitečný zejména pro ekonomické a finančních časové řady a předpovídání jejich budoucího vývoje [5]. Příkladem může být VAR(1) pro dvě proměnné X_{1t} a X_{2t}:

$$\begin{cases} X_{1t} = \varphi_{11} X_{1,t-1} + \varphi_{12} X_{2,t-1} + w_{1t} \\ X_{2t} = \varphi_{21} X_{1,t-1} + \varphi_{22} X_{2,t-1} + w_{2t} \end{cases} \tag{2.3}$$

kde konstanty φ_{ij} určují vliv jednotlivých zpožděných hodnot a $w_{1t,2t}$ jsou hodnoty bílého šumu.

Predikce pomocí nelineárních modelů

Jestliže časová řada reprezentuje nějaký nelineární jev, například populační cyklus zvířat, kde predikovaná hodnota vykazuje nelineární závislost na předchozí hodnotě, pak může být použití lineárního modelu nevhodné z důvodu velké nepřesnosti. Pro modelování nelineárních časových řad existují speciální modely jako Markovův přepínací model nebo modely ARCH a GARCH.

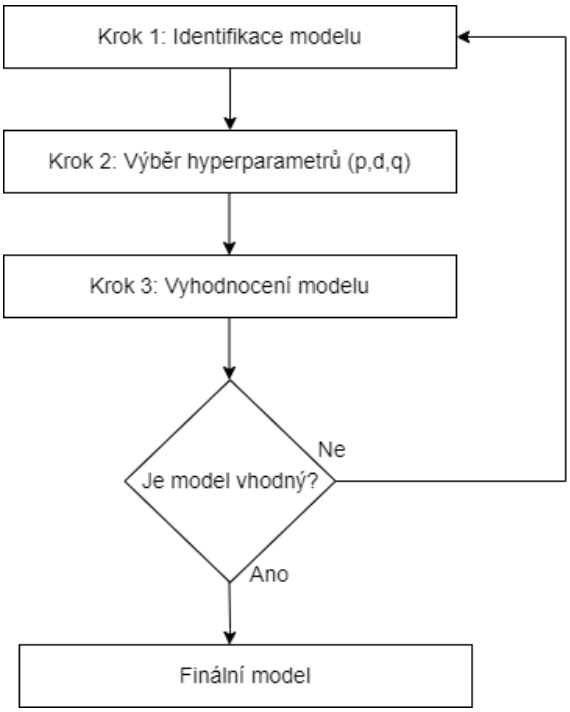
V případě predikce budoucího vývoje řady však použití nelineárního modelu nemusí znamenat lepší přesnost. Pokud použití lineárního modelu pro predikci nevykazuje mnohem větší chybovost než nelineární model, je lepší z důvodu zachování principu co největší jednoduchosti použít lineární model [10]. Ten je také obecně méně náročnější na výpočetní zdroje. Před aplikací nelineárního modelu na časovou řadu je dobré provést test nelinearity z důvodu nepravděpodobně dobrého přizpůsobení jakémukoli souboru dat časových řady. To se provádí nejčastěji testem BDS [22].

2.4.2 Box-Jenkinsův proces

Při pokusu o analýzu časové řady může nastat problém, a sice jak vybrat vhodný model, který bude řadu dobře popisovat, zároveň bude schopen poskytovat přesnou předpověd na základě historického vzorce v datech a tím pádem určovat optimální řád modelu.

Box-Jenkinsův proces je praktický postup vedoucí k sestavení modelu ARIMA, jež bude nejlépe odpovídat dané časové řadě a zároveň splňovat princip co největší jednoduchosti. Proces nepředpokládá žádný konkrétní vzorec v historických datech řady, která má být predikována.

Proces funguje na principu třístupňové iterace, kdy se v prvním kroku vybere vhodný model, následně odhadnou parametry modelu a nakonec se zkontroluje adekvátnost použitého modelu, tedy s jakou přesností dokáže model předpovídat časovou řadu. Tento tříkrokový postup se několikrát opakuje, dokud není nakonec vybrán nejlepší možný model [15].



Obrázek 2.3: Box-Jenkinsův proces.

2.4.3 Autokorelační funkce (ACF)

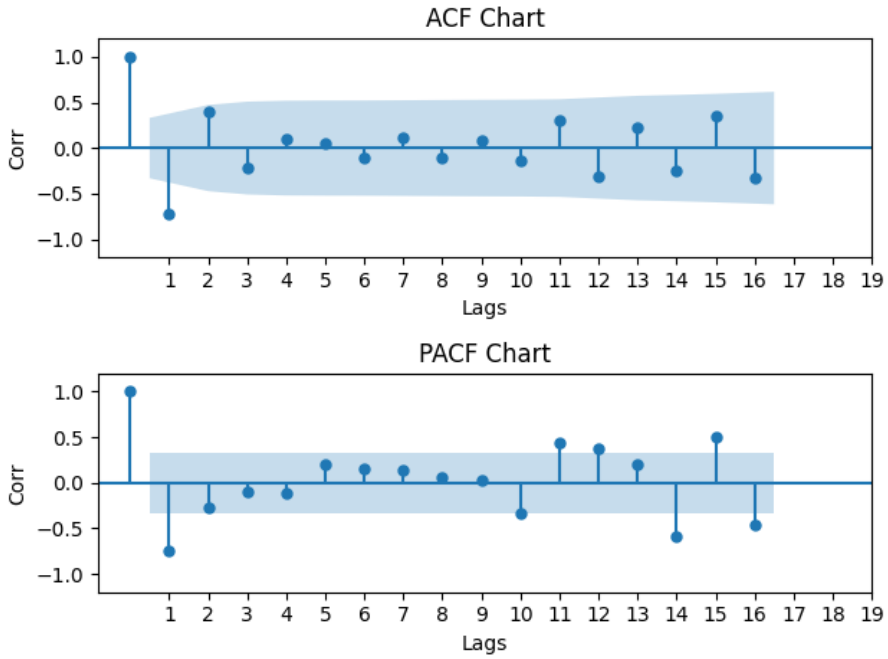
Autokorelační funkce ukazuje míru podobnosti časové řady s její zpožděnou verzí v po sobě jdoucích časových intervalech. Jinak řečeno, ACF vyjadřuje lineární vztah mezi hodnotou časové řady X_t a všemi minulými hodnotami X_{t-n}. Zjištění takového vztahu je důležité, protože kdyby byl na zkoumanou řadu aplikován regresní model, např. AR(p), a mezi hodnotami existovala autokorelace, tak by výsledky modelu byly značně zavádějící [8].

ACF také umožňuje ukázat na skryté vzorce v hodnotách časové řady, které by prostým vykreslením řady do grafu nemusely být na první pohled dobře viditelné.

2.4.4 Parciální autokorelační funkce (PACF)

Parciální autokorelace je souhrnný vztah mezi hodnotou v časové řadě a hodnotami v předchozích časových krocích s odstraněním vztahů mezilehlých hodnot (tj. s vyloučením korelačních příspěvků z mezilehlých zpoždění).

Parciální autokorelace se zpožděním n je korelace, která vznikne po odstranění vlivu případných korelací způsobených hodnotami řady, které mají kratší zpoždění (tedy zpoždění menších jak n). Vykreslení grafu PACF pak pomáhá určit ideální řád pro autoregresní model. Je dobré poznamenat, že jak graf autokorelační funkce, tak i parciální autokorelační funkce by měl být vykreslován na stacionární časové řadě. V opačném případě nebude získaná informace o nejoptimálnějším řádu pro model klouzavého průměru, resp. pro autoregresní model příliš vypovídající.



Obrázek 2.4: Graf autokorelační a parciálně autokorelační funkce.

2.4.5 Akaikeho informační kritérium

Akaikeho informační kritérium (AIC) je měřítkem relativní kvality statistického modelu pro daný soubor dat. Používá se pro výběr modelu a vychází z pravděpodobnostní funkce a složitosti modelu. AIC poskytuje způsob, jak kvantifikovat relativní kvalitu modelu tím, že bere v úvahu jak shodu modelu s daty, tak složitost samotného modelu [4]. AIC pomáhá určit, který model nejlépe odpovídá datům a je nejvhodnější pro zkoumanou časovou řadu. Může také určit, zda v datech nebo v samotném modelu nejsou nějaké chyby nebo nedostatky.

2.4.6 Bayesovské informační kritérium

Bayesovské informační kritérium (BIC) je kritérium pro výběr modelu z konečného souboru modelů. Vychází z myšlenky, že nejlepší model je ten, který nejlépe vyvažuje přizpůsobení se datům a složitost. BIC přiřadí každému modelu skóre a model s nejnižším skóre je vybrán jako nejlepší. BIC penalizuje složitost, což znamená, že složitější modely budou mít vyšší skóre než ty jednodušší, a tudíž bude menší pravděpodobnost, že budou vybrány jako nejlepší možný model.

2.4.7 Metriky pro výpočet přesnosti predikce

Při posuzování přesnosti jakéhokoli prognostického modelu je třeba vyhodnotit přesnost předpovědi, kterou model poskytuje. To se provádí výpočtem vhodných metrik chyb.

Metrika chyb je způsob, jak zjistit přesnost modelu, a poskytuje možnost pro porovnání různých modelů. Existují různé hodnotící metriky jejichž výpočet poskytuje způsob, jak objektivně posoudit, zda je předpovědní model vhodný pro danou časovou řadu:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|(y_t - \acute{y}_t)| \tag{2.4}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \acute{y}_t)^2 \tag{2.5}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \acute{y}_t)^2} \tag{2.6}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}|\frac{y_t - \acute{y}_t}{y_t} * 100| \tag{2.7}$$

kde y_t odpovídá aktuální hodnotě a $\acute{y}_t$ predikované hodnotě. n pak odpovídá počtu předpovězených hodnot.

Rovnice (2.4) reprezentuje snadno vypočítatelnou střední absolutní chybu (MAE). Naproti tomu, střední čtvercová chyba (MSE) (2.5), stejně tak jako odmocnina ze střední čtvercové chyby (RMSE) (2.6) dává mnohem větší důležistost výraznějším odchylkám [17]. Poslední rovnice popisuje střední absolutní procentní chybu (MAPE) (2.7), kterou lze snadno interpretovat. Jednou z jejích nevýhod však je, že generuje nekonečné či nedefinované hodnoty pro hodnoty y_t, jež jsou nulové nebo blízké nule [12].

2.5 Předpovídání vývoje pomocí neuronových sítí

Použití neuronové sítě pro predikci a modelování časové řady je odlišný, ale zcela životaschopný přístup ve srovnání s různými tradičními statistickými modely. Modely založené na neuronových sítích jdou poměrně dobře aplikovat na nelineární nebo složité řady, kde klasické statistické modely mohou selhat, jak je popsáno v 2.4.1.

2.5.1 Neuronová síť

Neuronová síť je technologie, která se snaží matematicky namodelovat způsob, jakým přemýšlí lidský mozek. Lze si ji představit jako systém různých spojení mezi množinou vstupů a množinou výstupů. Tato spojení jsou typicky tvořena jednou či více skrytými vrstvami neuronů, kde se každý z nich připojuje k jinému neuronu a má přiřazenou svou váhu a aktivační funkci. Jestliže je výstupní hodnota některého z neuronů vyšší než zadaná prahová hodnota (případ binární aktivační funkce), dojde k jeho aktivaci, což má za následek propagaci dat do další vrstvy. V opačném případě nejsou zpropagována žádná data do další vrstvy.

Existuje mnoho různých architektur neuronových sítí a aplikace správného druhu či zvolení ideálního počtu skrytých vrstev závisí zejména na druhu řešené problematiky a dostupné datové sadě.

2.5.2 Bázová funkce

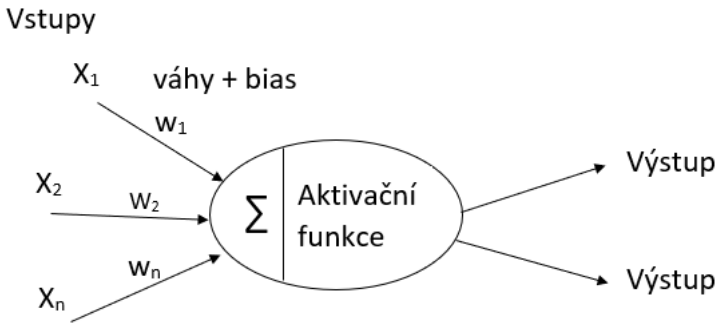
Bázová funkce slouží k převodu množiny vstupů neuronu na jedinou hodnotu g(x). Tato celková sumace je zároveň hodnota, jež vstupuje do aktivační funkce. Bázová funkce se vypočítá jako:

$$g(x) = bias + \sum_{i=1}^{n} X_i w_i) \tag{2.8}$$

kde X_i jsou jednotlivé vstupy a w_i jsou váhy přiřazené jednotlivým vstupům. Poslední složkou rovnice je bias, což je určitá konstanta, umožňující posun křivky aktivační funkce. Bias také pomáhá modelu neuronové sítě se lépe přizpůsobit datové sadě.

2.5.3 Aktivační funkce

Hlavním účelem aktivační funkce je rozhodnout, jestli je vstupní vrstva neuronu důležitá či nikoliv. Visualizace přenosu vstupní vrstvy na výstup je znázorněna na obrázku 2.5. Samotná celková hodnota výstupní vrstvy se pak vypočítá jako:



Obrázek 2.5: Neuron se vstupní vrstvou, výstupní vrstvou a aktivační funkcí

$$Y(x) = \phi(g(x)) \tag{2.9}$$

kde $g(x)$ je bázová funkce popsaná výše a ϕ je samotná aktivační funkce. Tabulka 2.1 ukazuje nejznámější aktivační funkce.

Sigmoid, známý také jako logistická funkce nabývá hodnot od 0 do 1 a využívá se pro předpoveď pravděpodobnosti [19]. Hyperbolický tangens je aktivační funkce velice podobná sigmoidu s tím rozdílem, že má strmější křivku a její obor hodnot je od -1 do 1. Aktivační funkce ReLU (Rectified Linear Unit) zase umožňuje ignorovat neurony, které nejsou důležité pro prediktivní sílu modelu [21].

2.5.4 Datová sada

Jedním z nejdůležitějších předpokladů při použití neuronové sítě na vytváření predikcí, je existence kvalitní a dostačně obsáhlé datové sady. Neuronová síť je technologie velmi závislá na datech a bez kvalitního a reprezentativního souboru dat nelze dost dobře vytvořit použitelný prediktivní model [13].

Nutnost dostatečně obsáhlého souboru dat vzniká také proto, že datovou sadu je potřeba rozdělit na dvě části: trénovací a testovací. Pokud by byla celá datová sada použitá jen

Název	Funkce	Derivace
Sigmoid	$\phi(x) = \frac{1}{1+e^{-x}}$	$\phi'(x) = \phi(x)(1-\phi(x))$
TanH	$\phi(x) = \frac{2}{1+e^{-2x}} - 1$	$\phi'(x) = 1 - \phi(x)^2$
Binární funkce	$\phi(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$	$\phi'(x) = \begin{cases} 0 & x \neq 0 \\ ? & x = 0 \end{cases}$
ReLU	$\phi(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$	$\phi'(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$

Tabulka 2.1: Nejznámější aktivační funkce

k trénování neuronové sítě, mohl by vzniknout problém nadměrného přizpůsobení, lépe známým pod anglickým slovem overfitting. Rozdělení souboru dat na dvě části se dělá obvykle v poměru 70:30 nebo 80:20, u malých datových sad alespoň poměrem 90:10 [13].

2.5.5 Učení s učitelem vs. učení bez učitele

Učení s učitelem a učení bez učitele jsou dva různé typy algoritmů strojového učení. Při použití algoritmu učení s učitelem se používají označená data k vytvoření predikčního modelu. Označenými daty se myslí soubor vstupních a výstupních hodnot, které jsou používány k trénování modelu. Ten se pak používá k vytváření předpovědí o nových datech. Cílem učení s učitelem je na základě označených trénovacích dat přesně předpovědět výstupní hodnoty nových dat.

Naproti tomu, učení bez učitele k vytvoření modelu používá neoznačená data, kdy se model trénuje na neoznačených datech identifikováním vzorů a podobností v datech. Cílem učení bez učitele je identifikovat v datech smysluplné vzory a struktury [20].

2.5.6 Rekurentní neuronová síť

Rekurentní neuronová síť (RNN) je typem neuronové sítě, jež si uchovává paměť toho, co již zpracovala, a může se tak během svého trénování učit z předchozích iterací. Vytváří rekurentní spojení tak, že prochází časovými smyčkami zpětné vazby, kde výstup předchozího kroku je použit jako vstup pro aktuální krok. Na rozdíl od dopředné neuronové sítě (FFNN) tato paměť umožňuje RNN zpracovávat sekvence vstupů, aniž by ztratila přehled.

Skryté vrstvy, které se nacházejí mezi vstupní a výstupní vrstvou nejenže vytvářejí výstup, ale také jej vracejí zpět jako vstup pro trénování následující skryté vrstvy. Trénování je prováděno nastavováním vah synapsí v celé neuronové síti. Síť překalibruje váhy pro aktuální i předchozí vstupy, vynásobí vektor vstupních hodnot vektorem nových vah, čímž zvýší nebo sníží jejich důležitost vzhledem k tréninkovému cíli, kterým je snížení chyby predikce, a předá vektor výsledků jako vstup do další vrstvy.

Přizpůsobováním vah skrytá vrstva postupně odvozuje funkci, která transformuje vstupní hodnoty na výstupní hodnoty, jež se blíží skutečným hodnotám v trénovací množině dat. Funkce, která mapuje vstupy na výstupy, však není vyjádřena jako matematická rovnice,

ale zůstává skrytá. Když je vstupní hodnota předávána ze svého uzlu v jedné vrstvě do uzlu v jiné vrstvě, tak se pohybuje po hraně mezi uzly.

Přijímající uzel po obdržení všech vstupů provede sumaci do celkového vstupu. Tento celkový vstup vloží do aktivační funkce, která vypočítá výstup, tedy kolik bude uzel přispívat do další vrstvy. Nelineární aktivační funkce, jako je logistická funkce nebo funkce hyperbolického tangensu, pomáhají síti přizpůsobit se nelineárním problémům při mapování vstupu na výstup trénování. Výstupní hodnota aktivační funkce se zároveň násobí váhovým faktorem, když je vyslána po hraně do uzlu v jiné vrstvě nebo po více hranách do více než jednoho přijímacího uzlu.

Dlouhá krátkodobá paměť

Dlouhá krátkodobá paměť (Long short-term memory, LSTM) je rozšíření RNN, které umožňuje uchovávat dlouhodobou paměť a využívat ji k učení v delších sekvencích zdrojových dat. Dokáže si uchovat dlouhodobé vzory, které objeví při průchodu svými smyčkami. V každém časovém kroku může spojit tři druhy informací, a sice aktuální vstupní data, krátkodobou paměť obdrženou od předchozí buňky (tzv. skrytý stav), a dlouhodobou paměť od vzdálenějších buněk (tzv. stav buňky), z níž buňka RNN vytvoří nový skrytý stav.

Rozšíření LSTM také řeší problémy zvané mizivý sestup gradientu a problém explodujícího gradientu [14]. V prvním případě se model přestane učit, protože sklon gradientu se stane příliš mělkým na to, aby hledání umožnilo další zlepšování vah. K tomu dochází, když existuje velký počet hodnot zapojených do opakovaných výpočtů gradientu menších než 1. Problém mizejícího gradientu se může objevit, pokud se předpovědní model musí vypořádat s dlouhými časovými řadami [1]. LSTM udržuje gradienty dostatečně strmé, aby hledání neuvízlo ve slepé uličce. Explodující gradient zase způsobuje nestabilitu RNN opakovaných násobeních matic, když mnoho hodnot přesahuje 1.

O buňce v LSTM se říká, že je "gated". Informace se selektivně přidávají nebo odebírají prostřednictvím bran. Buňka funguje jako síto, které určuje, kolik příchozích informací se zachytí a kolik se jich zachová. Model se může rozhodnout, zda otevře vstupní bránu pro uložení informace, buď ji odmítne a odstraní z dlouhodobé paměti (brána pro zapomenutí), nebo informaci předá další vrstvě (výstupní brána). RNN činí tato rozhodnutí na základě vah důležitosti, které se naučí přiřazovat informacím, když se snaží minimalizovat chybu a prochází svými časovými smyčkami. Brány provádějí maticové násobení mezi hodnotami, které dostávají jako své aktuální vstupy, z krátkodobé nebo dlouhodobé paměti. V průběhu času se LSTM naučí, které části informací jsou účinné pro snížení chyby předpovědi a podle toho bude otevírat a zavírat brány tím, že hodnotám přiřadí vyšší nebo nižší váhy z rozsahu 0 až 1. Prostřednictvím svých smyček nechá užitečné hodnoty s vyššími váhami projít výstupním hradlem a vytvoří novou krátkodobou paměť, ale hodnoty s nízkou váhou vyřadí.

2.5.7 Konvoluční neuronová síť

Konvoluční neuronová síť (CNN), je biologicky inspirovaný typ hluboké neuronové sítě, která si získala popularitu zejména díky úspěchu v klasifikačních problémech (např. rozpoznávání obrazu). CNN se skládá ze sledu konvolučních vrstev, jejichž výstup je připojen pouze k lokálním oblastem na vstupu. Tato struktura umožňuje modelu naučit se filtry, které jsou schopny rozpoznat specifické vzory ve vstupních datech.

Lokální konektivita

Konvoluční neuronové sítě byly vyvinuty s myšlenkou lokální konektivity. To znamená, že každý uzel je připojen pouze k lokální oblasti na vstupu. Lokální konektivity je dosaženo nahrazením vážených součtů z neuronové sítě konvolucemi. V každé vrstvě konvoluční neuronové sítě je vstup konvolvován s váhovou maticí (nazývanou také filtr), čímž se vytvoří mapa příznaků. Jinými slovy, váhová matice přechází přes vstup a vypočítává skalární součin mezi vstupem a váhovou maticí.

Na rozdíl od běžných neuronových sítí mají všechny hodnoty ve výstupní mapě příznaků stejné váhy. Tím pádem všechny uzly na výstupu detekují přesně stejný vzor. Aspekt lokální konektivity a sdílených vah u CNN snižuje celkový počet učitelných parametrů, což vede k efektivnějšímu tréninku.

Kapitola 3

Návrh řešení

Tato kapitola pojednává o návrhu a architektuře řešení, které bude implementováno.

3.1 Cíl práce

Cílem této práce je navrhnout program pro předpovídání časových řad s využitím algoritmů strojového učení, který by načítá data časových řad a předpovídat hodnoty v nich v zadané časové vzdálenosti. Také musí existovat možnost vyhodnotit kvalitu navržených a nakonfigurovaných modelů. Tento program je také potřeba implementovat a následně otestovat na jednoduchých a složitějších vícedimenzionálních datech za účelem získání co nejpřesnějších výsledků předpovídání.

3.2 Architektura

Samotný program je navržen tak, aby uživateli umožnil co nejjednodušší možnost práce s analýzou a předpovídáním časových řad. Řešení je rozděleno na grafickou část vytvořenou v programu Qt Designer 4.1.2 a funkcionální část, zodpovědnou za samotné načítání a analýzu řad, stejně tak jako za trénování a vyhodnocování jak statistických modelů, tak i modelů neuronových sítí. Obě části řešení jsou navrženy takovým způsobem, aby bylo umožněno snadné rozšíření programu o nové předpovědní modely. Program je koncipovaný jako knihovna jazyka python, je tedy možné si jej nainstalovat pomocí nástroje pip.

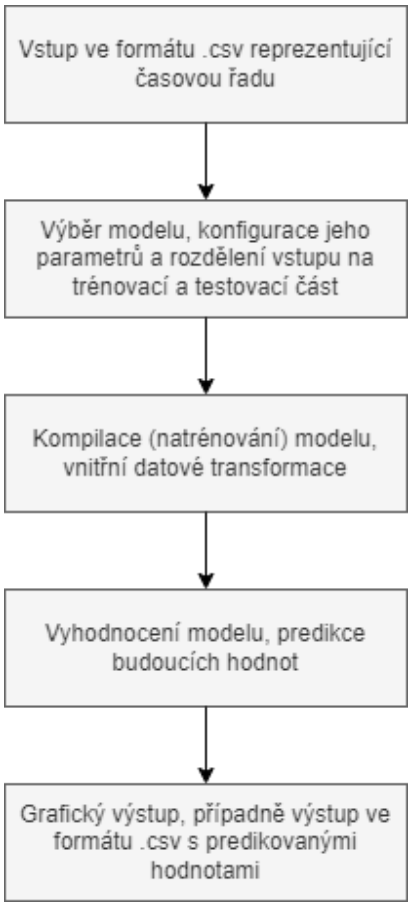
Diagram, zobrazující vztahy mezi jednotlivými třídami je na obrázku 3.2.

3.3 Popis funkcionality

Vstupem je soubor ve formátu .csv, který na prvním řádku obsahuje hlavičku se jmény sloupců. První sloupec slouží jako index, měl by tedy obsahovat datum. Ostatní sloupce reprezentují samotná data časové řady, kde jeden řádek odpovídá jednomu pozorování v čase. Po spuštění programu si uživatel vybere soubor s časovou řadou, na kterém chce provádět analýzu. Tento soubor se pro účely dalšího zpracování převede do objektu Dataframe z knihovny pandas.

Poté uživatel zvolí jeden z implementovaných modelů a nastaví jeho parametry. Pro účely vyhodnocení použitelnosti modelu s danými parametry si také nastaví vhodný poměr rozdělení souboru s řadou na trénovací a testovací část. Následně patřičně nakonfigurovaný model zkompiluje. Datové úpravy a transformace se, pakliže je vybraný model vyžaduje,

provádí automaticky. Nakonec si analytik časové řady může vybraný a natrénovaný model vyhodnotit na rozdělené datové sadě, jež si nastavil před vyhodnocením modelu. Postup zpracování časové řady od vstupu na výstup znázorňuje blokové schéma 3.1.



Obrázek 3.1: Blokové schéma, popisující zpracování vstupního souboru s časovou řadou

Plánované funkcionality

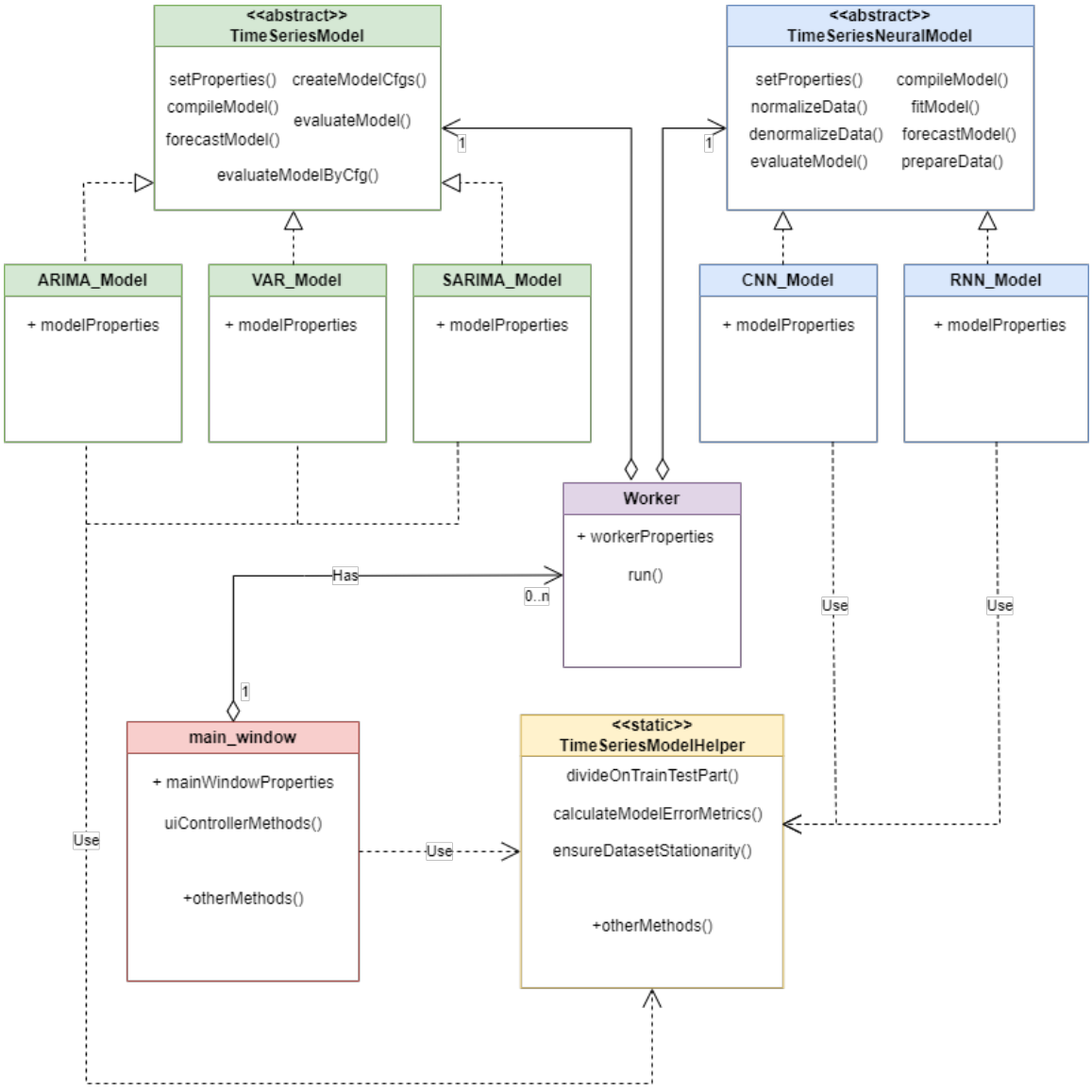
Mezi základní výčet funkcionalit bude patřit:

- Načtení datasetu a jeho grafické zobrazení.
- Kontrola stacionarity, zobrazení grafu autokorelační a parciální autokorelační funkce.
- Zvolení vhodného modelu, nastavení jeho parametrů a jeho vyhodnocení na datasetu rozděleném na trénovací a testovací část.
- Zobrazení grafických výstupů a chybových metrik.
- Nalezení co možná nejlepšího nastavení hyperparamerů u statistických modelů.
- Předpovídání budoucích hodnot jedné nebo více proměnných v závislosti na použitém modelu.
- Uložení nakonfigurovaného nebo natrénovaného modelu tak aby uživatel nemusel tento proces v budoucnu opakovat.

3.4 Návrh rozhraní modelů

Každý použitý model, jedno jestli je statistický nebo obsahuje neuronovou síť, bude implementovat jednotné rozhraní. To je výhodné z toho důvodu, že umožňuje program snadno rozšířit o další modely, aniž by programátor musel zdlouhavě zkoumat zdrojový kód celého programu.

Rozhraní budou dvě, jedno pro statistické modely a druhé pro modely s neuronovou sítí. Hlavní rozdíl mezi těmito dvěmi je ten, že rozhraní pro neuronové modely bude obsahovat metody pro normalizaci dat, přípravu dat a metodu pro samotné natrénování neuronové sítě, kde její učení trvá násobně déle než učení u statistických modelů.



Obrázek 3.2: Diagram tříd, znázorňující navrženou architekturu

Kapitola 4

Implementace

Tato kapitola pojednává o implementaci samotného programu a o problémech, které se při řešení vyskytly.

4.1 Použité technologie

Technologie, programovací jazyky, frameworky a knihovny, které budou použity při implementaci programu.

4.1.1 Python

Python je vysokoúrovňový, interpretovaný, univerzální programovací jazyk, který se v posledních letech stále častěji používá k různým aplikacím v oblasti vývoje softwaru, skriptování, tvorby skriptů, umělé inteligence a datové vědy. Jazyk má jednoduchou syntaxi a snadno se učí, takže je oblíbený jak mezi začínajícími, tak zkušenými vývojáři.

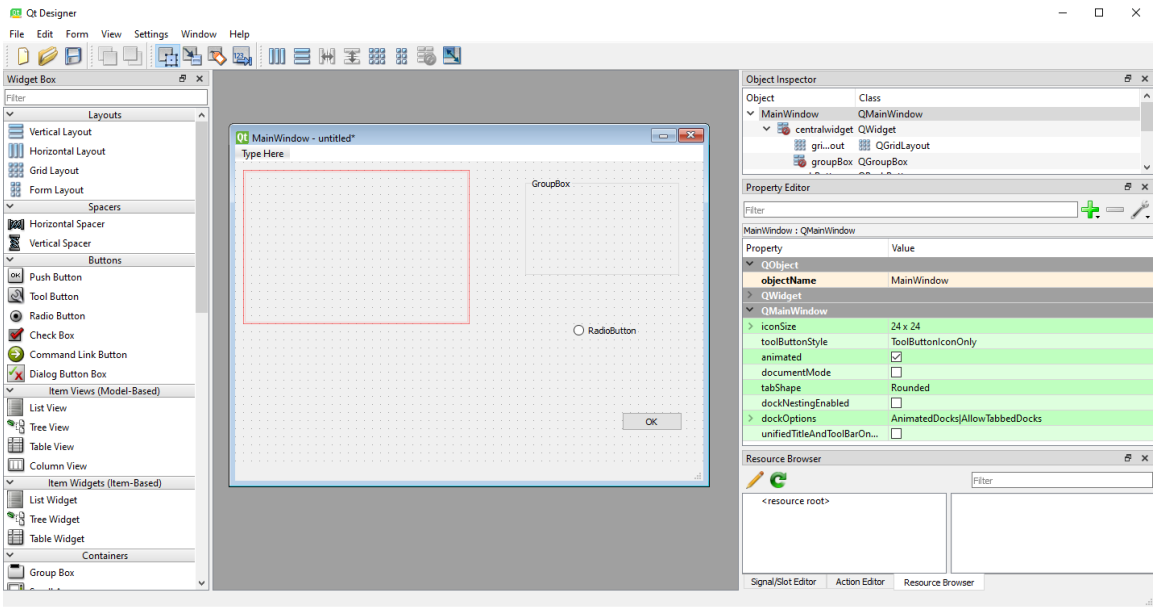
Python také disponuje širokou škálou knihoven a frameworků. Mezi oblíbené knihovny patří knihovna pro vědecké výpočty SciPy či knihovna pro numerické výpočty NumPy. Existuje též značný počet knihoven pro strojové učení a tvorbu neuronových sítí, stejně tak pro práci s daty. Tyto knihovny poskytují vývojářům výkonné nástroje pro analýzu dat, strojové učení. Je schopen snadno zpracovávat velké soubory dat a složité algoritmy, což vývojářům umožňuje vytvářet komplexní aplikace.

4.1.2 PyQt

PyQt je populární knihovna jazyka Python pro tvorbu grafických uživatelských rozhraní (GUI). Je založena na Qt, výkonném multiplatformním aplikačním rámci. Poskytuje ucelenou sadu knihoven a nástrojů pro tvorbu uživatelských rozhraní. Patří mezi ně např. QtCore, QtWidgets nebo QtGui. Tyto mají širokou škálu funkcí, například podporu bohatého textu a grafiky, animace, přetahování, dialogy a další. Poskytují také širokou škálu widgetů, jako jsou tlačítka, štítky, textová pole a zaškrtávací políčka. Jsou obsaženy i možnosti pro vytváření vlastních widgetů a dialogových oken.

Výhodou tohoto frameworku je také existence vývojových nástrojů jako Qt Designer či Qt Creator. Nástoj Qt Designer bude použit při implementaci grafického uživatelského rozhraní.

PyQt je výkonná knihovna pro vytváření robustních multiplatformních aplikací. Je vhodná pro vývoj komerčních i open-source aplikací. Jedná se o vyspělou knihovnu, která je dobře zdokumentovaná a má aktivní komunitu vývojářů.



Obrázek 4.1: Nástroj Qt Designer, který bude použit pro deklaraci GUI

4.1.3 Keras

Keras je knihovna pro hluboké učení v jazyce Python, která poskytuje vysokoúrovňové aplikační rozhraní (API) pro vytváření a trénování neuronových sítí. Jedná se o populární open-source framework se zaměřením na možnost rychlého experimentování s modely hlubokého učení.

Keras poskytuje snadno použitelné rozhraní pro konstrukci a trénování neuronových sítí, aniž by bylo nutné rozumět detailům základních algoritmů. Poskytuje konzistentní sadu funkcí pro sestavování a trénování modelů v různých architekturách, jako jsou konvoluční neuronové sítě, rekurentní neuronové sítě a další. Keras navíc umožňuje uživatelům rychle definovat a trénovat modely s minimálním množstvím kódu [3].

4.1.4 Statsmodels

Statsmodels je knihovna jazyka Python pro statistické modelování a analýzu dat. Je postavena nad populární knihovnou SciPy a poskytuje řadu funkcí pro analýzu dat a statistické testování. Statsmodels se hodí zejména pro analýzu časových řad, protože poskytuje širokou škálu funkcí pro manipulaci s daty v časových řadách, jejich analýzu a předpovídání [18].

Knihovna obsahuje kolekci specializovaných podmodulů pro analýzu univarietních a multivarietních časových řad. Poskytuje také širokou škálu statistických testů, jako je testování hypotéz, regrese a stacionarity. Tyto testy jsou užitečné pro pochopení základních vlastností dané časové řady. Knihovna rovněž poskytuje funkce pro provádění různých typů prognóz, včetně exponenciálního vyhlazování, modelů ARIMA a SARIMA či vektorové autoregrese (VAR).

4.2 Popis programu

Jádrem programu je třída MainWindow. Tato slouží jako kontrolér pro grafické uživatelské rozhraní (GUI), vytvořené v rámci frameworku PyQt 4.1.2 a zároveň obsahuje všechny instance aktuálně používaných modelů. Vzhledem k tomu, že hlavní okno běží v rámci jednoho procesu, není možné v rámci zachování responzivity v tom samém procesu provádět časově náročné operace, jako je např. trénování neuronových sítí nebo vyhodnocování kvality modelů na rozsáhlých datasetech. K vyřešení tohoto problému slouží třída Worker, která pro všechny časově náročné operace vytváří asynchroní proces.

Pro vykreslování výstupních grafů, jakožto i další pomocné činosti, slouží statická třída TimeSeriesModelHelper. Abstraktní třídy TimeSeriesModel a TimeSeriesNeural slouží jako rozhraní pro sadu metod, jež následně implementují třídy jednotlivých modelů, kde jedna třída odpovídá jednomu modelu. Popis jednotlivých metod obou rozhraní je v tabulce 5.4.

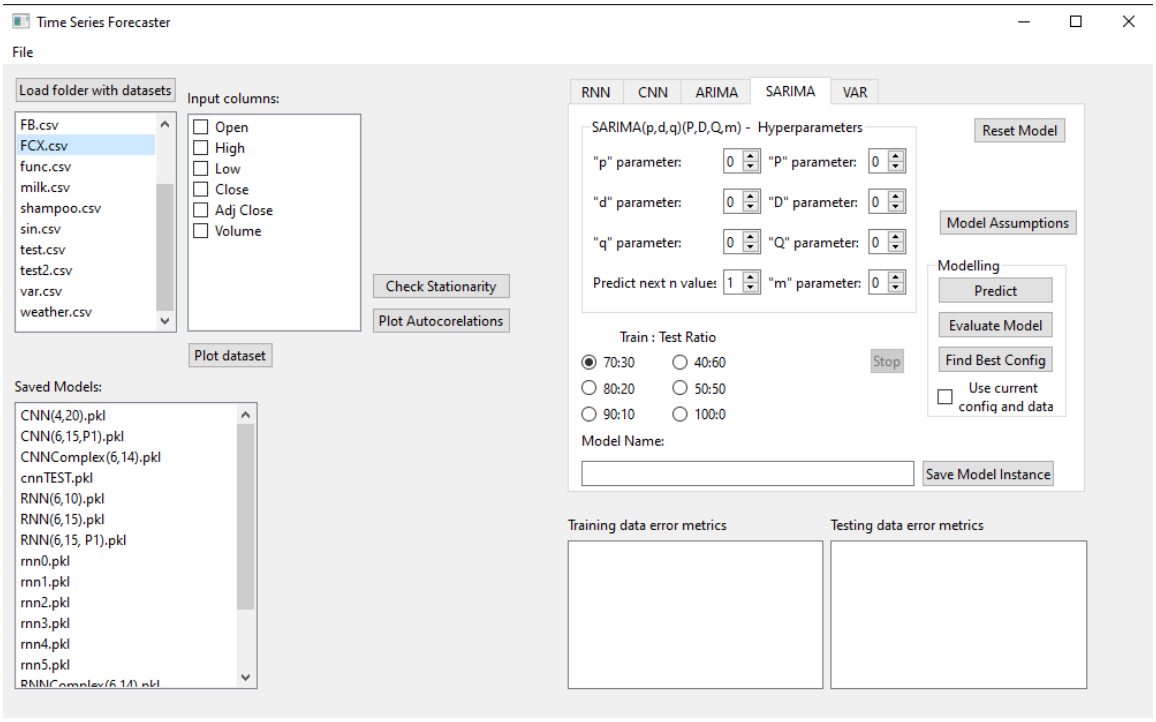
TimeSeriesNeuralModel	TimeSeriesModel	Popis funkce
setProperties()	setProperties()	Slouží pro nastavení počátečního vstupního datasetu, parametrů modelu a poměru rozdělení datsetu na trénovací a testovací část.
compileModel()	compileModel()	Funkce pro inicializace modelu se zvolenými parametry.
evaluateModel()	evaluateModel()	Funkce pro vyhodnocení kvality modelu se zvolenými parametry na datasetu, rozděleném na trénovací a testovací část.
forecastModel()	forecastModel()	Funkce pro predikci neznámých, budoucích hodnot časové řady.
normalizeData()		Slouží pro normalizaci dat na specifikovaný rozsah hodnot pro modely s neuronovou sítí.
denormalizeData()		Slouží pro denormalizaci dat na původní hodnoty vstupní časové řady pro modely s neuronovou sítí.
prepareData()		Finální příprava dat (např. učení s učitelem), na těchto datech se bude síť učit.
fitModel()		Po zavolání této funkce se neuronová síť začne učit.
	evaluateModelByCfg()	Slouží pro vyhodnocení statistického modelu s aktuálně zvolenými hyperparametry
	createModelCfgs()	Funkce pro vytvoření použitelných kombinací hyperparametrů.

Tabulka 4.1: Popis jednotlivých metod rozhraní

4.3 Uživatelské rozhraní

Grafické uživatelské rozhraní je hlavní částí celého programu. Implementoval jsem jej jako jednoúrovňové, s ohledem na co nejsnažší použitelnost. Lze rozdělit na dvě hlavní části. První část je společná pro všechny druhy modelů a sestává z okna pro načítání souborů se vstupními řadami. Jakmile si uživatel vybere vstupní soubor, v okně napravo se mu zobrazí možnost výběru jednoho nebo více sloupců, kde každý sloupec reprezentuje jednu proměnnou časové řady. Vedle těchto dvou oken se nalézají tlačítka pro vynesení zvolených sloupců časové řady na graf, kontrolu stacionarity a zobrazení grafu autokorelace a parciální autokorelace. Sluší se podotknout, že pokud kontrolovaná proměnná není stacionární, automaticky se před zobrazením těchto dvou grafů stacionarizuje. Ve spodním okně se pak nabízí možnost načíst uložené nebo natrénované modely. Poslední společnou komponentou pro všechny modely jsou okna pro zobrazení chybových metrik pro vyhodnocování jednotlivých modelů na trénovacích a testovacích datech.

Druhá část rozhraní je implementována jako okno se záložkami, přičemž každá záložka odpovídá jednomu modelu. Důvod rozdělení na záložky je ten, že každý model má svoje specifické nastavení, které závisí od jeho vnitřních vlastností. Zároveň umožňuje toto řešení přidávat další modely bez nutnosti rozsáhlých změn celého rozhraní. Na jednotlivých záložkách se nastavují vstupní parametry nebo hyperparametry a poměrové rozdělení dat na trénovací a testovací část. Tlačítka pro ovládání veškerých operací jako je kompilace modelu, vyhodnocení kvality, predikce budoucích hodnot a v neposlední řadě tlačítko pro nalezení nejlepšího nastavení hyperparametrů, se nachází také zde.



Obrázek 4.2: Výsledné GUI

4.3.1 Komunikace mezi objekty

V knihovně PyQt existuje systém slotů a signálů, jenž umožňuje objektům vzájemně komunikovat bez jejich přímého propojení. Sloty jsou metody a funkce, které lze připojit k signálům vysílaným z jiných součástí aplikace. Signály jsou jednoduše události, které se spouštějí, když něco nastane.

Když je signál odeslán, bude připojen ke všem slotům, které jsou pro signál relevantní, což umožní, aby se při výskytu události provedla konkrétní akce nebo sada akcí. Aby bylo možné systém slotů a signálů používat, musí třída nejprve vytvořit objekt signálu nebo slotu a poté jej připojit k signálu nebo slotu jiné třídy pomocí metody PyQt.QObject.connect(). Vyslání signálu se dělá obdobně, a sice pomocí metody emit() zavolané na objektu signálu.

4.3.2 Třída Worker

Pro správnou funkčnost celého programu je nutné vyřešit zpracování asynchronních operací. Trénování neuronových sítí nebo vyhodnocování modelů na rozsáhlých časových řadách může být časově velmi náročné a tudíž není možné těmito operacemi blokovat hlavní proces, který obsluhuje uživatelské rozhraní. Tento problém řeší třída Worker.

Jedná se o objekt, sloužící k vytváření pomocných pracovních vláken pro dlouhotrvající operace. Worker obsahuje tři druhy signálů: result, error a finished. Signál result se vyšle pokud je úspěšně dokončena nějaká dlouhotrvající operace a je potřeba spustit na ni navázanou další logiku. Signál error slouží k vyvolání okna s chybovou hláškou a informování uživatele o nějaké nastanuvší chybě. Poslední ze signálů, finished, se vyvolá vždy, nehledě na to zda operace skončila úspěchem či nikoli. Slouží typicky k opětovné aktivaci tlačítka, jež bylo dříve kvůli spuštění nějaké činnosti zneaktivněno.

Použití třídy Worker je následovné: Pokaždé když uživatel spustí časově náročnou funkcionalitu jako je např. předpověd budoucích hodnot časové řady, vytvoří se nová instance Workeru, kde prvním parametrem je název funkce (v tomto případě metoda forecastModel()). Další parametry jsou volitelné, v tomto případě by to byl ještě počet požadovaných budoucích hodnot. Následně se výše popsané signály napojí na obslužné metody. Nakonec se celá instance s operací přidá do poolu vláken kde dojde k jejímu spuštění.

4.4 Předpovědní modely

V programu jsou implementovány dva druhy předpovědních modelů. První z nich, statistické modely (ARIMA, SARIMA a VAR), využívají tradiční lineární a nelineární metody k analýze a předpovědi budoucích hodnot ze souboru dat s časovými řadami. Fungují na základě matematických úvah, jako je autokorelace, křížová korelace, sezónnost, trendy a další aspekty dat. Výhodou této metody je, že statistické metody jsou snáze pochopitelné a interpretovatelné a lze je použít i na malých souborech dat. Nevýhodou je, že nemusí být schopny zachytit komplexní vzory v datech nebo se učit ze zašuměných dat.

Druhé jsou modely, které využívají neuronové sítě, čímž pádem využívají výhod technik strojového učení. Tyto modely se skládají ze sítě neuronů v několika vrstvách, které jsou vzájemně propojeny tak, že výstup jedné vrstvy neuronů může být přiveden do další vrstvy. Výhodou těchto modelů je, že se mohou učit složitější vzory než standardní modely a mohou se učit i ze zašuměných dat. Tyto modely však vyžadují mnohem více dat a výpočetního výkonu. Je také potřeba neuronovou síť nejprve natrénovat. U statistických modelů je sice také nutné provést trénování, na stejném datasetu je však jejich natrénování řádově rychlejší.

Hlavním rozdílem mezi modely používající statistické metody a modely které používají neuronové sítě, je úroveň složitosti. Statistické modely umožňují vytvářet poměrně rychlé predikce na jednoduchých datech, zatímco modely s neuronovou sítí se mohou naučit složitější vzorce, zato ale vyžadují více dat a výpočetního výkonu.

4.4.1 Vyhodnocování modelů

Všechny implementovné modely mají možnost vyhodnocení na datech, rozdělených na trénovací a testovací část. Jestliže byl model natrénován na celém datasetu a nejsou tedy k dispozici testovací data, proběhne vyhodnocení na trénovacích datech. V opačném případě dojde k vyhodnocení jak na trénovacích, tak i testovacích datech.

Způsob vyhodnocování se liší podle toho, zda je model neuronový nebo statistický. U prvně jmenovaného vyhodnocení probíhá tak, že se vezme z testovací sady n hodnot a model vrátí predikovanou hodnotu (n+1). Následně se vezme n-1 hodnot z testovací sady, za které se přiřadí predikovanoá hodnota z minulého kroku a model vrátí další hodnotu. Takto se pokračuje pro celou délku testovacího datasetu. U statistických modelů tímto způsobem vyhodnocovat nelze, neboť neumožňují po natrénování předpovídat hodnotu z jiné hodnoty. Na natrénovaném modelu se zavolá funkce `forecast(n)`, kde n odpovídá délce testovacího datasetu, a tato funkce následně vrátí n hodnot.

4.4.2 Předpovídání budoucího vývoje

U každého modelu je zakotvena možnost predikce budoucího vývoje časové řady. V případě neuronového modelu je potřeba nejprve provést natrénování se zvolenými parametry a poté si uživatel zvolí požadovaný počet budoucích hodnot, jež chce predikovat.

4.4.3 Ukládání modelů

Do programu byla zavedena možnost uložit natrénovaný a vyhodnocený model. Způsob ukládání se liší podle druhu modelu. V případě statistického je ukládána celá instance modelu, a to pomocí nástroje `pickle`[1] ze standardní knihovny jazyka python.

V případě neuronových modelů je používána funkce `save()` z knihovny keras, kde se ukládají hodnoty vah natrénovaného modelu do souboru s příponou .h5. V průběhu implementace bylo totiž zjištěno, že použití nástroje `pickle` pro serializaci instancí s objekty knihovny keras padá na neočekávané chyby. Toto bylo vyřešeno tím způsobem, že při ukládání neuronového modelu se z jeho instance vyjme samotný objekt obsahující váhové koeficienty natrénované neuronové sítě, který je uložen do již zmíněného formátu .h5. Pomocí nástroje `pickle` se poté serializuje zbytek instance s používanými datasety a vstupními parametry pro neuronovou síť. Celá instance je tedy uložena do dvou souborů. Při načítání modelu je pak nejprve načten soubor uložený pomocí `pickle`, který projde deserializací, vytvoří se z něj nová instance modelu, do které je nakonec načten pomocí funkce `load_model()` soubor s příponou .h5.

4.5 Statistické modely

V programu jsou implementovány tři druhy statistických modelů, a sice modely ARIMA, SARIMA a VAR. Model ARIMA je založen na myšlence, že minulé hodnoty řady lze použít

[1] Pickle slouží pro serializaci a deserializaci python objektů: https://docs.python.org/3/library/pickle.html

k předpovědi budoucích hodnot a využívá tři složky: autoregresní složku, integrovanou složku a složku klouzavého průměru. Autoregresní složka využívá minulé hodnoty k predikci současné hodnoty, integrovaná složka zohledňuje nestacionární trendy v datech a složka klouzavého průměru bere v úvahu náhodné výkyvy v datech.

Model SARIMA je v podstatě rozšíření modelu ARIMA, které zohledňuje sezónní trendy v datech. SARIMA používá stejné složky jako ARIMA, ale přidává ještě čtvrtou složku, která zohledňuje sezónní vzorce v datech. Tato čtvrtá složka se nazývá sezónní složka a pomáhá modelu identifikovat a zohlednit všechny pravidelné cyklické vzorce v datech. Model SARIMA je užitečný pro předpovídání dat, která mají sezónní trend. Posledním statistickým modelem je VAR, což je model používaný pro vícerozměrné předpovědi časových řad. Model VAR zohledňuje vztah mezi více proměnnými a lze jej tudíž použít k předpovídání budoucích hodnot více proměnných současně.

Každý z implementovaných modelů má své silné a slabé stránky, díky čemuž je každý model vhodný pro různé typy řad. ARIMA se používá pro krátkodobé prognózování, SARIMA se používá pro předpovídání dat se sezónními trendy a model VAR je vhodný pro modelování multivarietních časových řad.

4.5.1 Diferenciace

Diferenciace je důležitý proces používaný při předpovídání časových řad jako součást statistických modelů. Jedná se v podstatě o transformaci časové řady na novou řadu, která představuje rychlost změn v hodnotách původní řady. Tento proces je např. v modelu VAR nutný, protože tento model vyžaduje, aby data byla stacionární 2.3.

Pokud data nejsou stacionární, model není schopen přesně zachytit vztahy mezi proměnnými, což vede k nepřesným předpovědím. Vycházením z prvního rozdílu časové řady lze odstranit vliv trendu a sezónnosti a datovou řadu učinit stacionární. To se provádí odečtením předchozí hodnoty od aktuální hodnoty, a to pro všechny hodnoty řady, čímž se získá míra změny řady. Tento postup lze provést vícekrát, aby se dále odstranila zbývající nestacionarita.

Tím, že se vezme rozdíl řad, dojde zároveň k odstranění první hodnoty řady, neboť první hodnota nemá svého menšitele. Po provedené předpovědi na diferencovaných datech je provést zpětnou diferenciaci provedením matematické operace kumulativní součet na všech hodnotách diferencované řady, ke kterým se nakonec přičte první hodnota původní řady. Tímto způsobem dojde k získání původní časové řady.

V rámci programu se diferenciace provádí automaticky, a to buď nastavením hyperparametru "d"v případě modelu ARIMA, nebo hyperparametrů "d"či "D"u modelu SARIMA. U modelu VAR se diferenciace automaticky provádí na základě výsledku testu ADF, popsaného v sekci 2.3.

4.5.2 Vyhledávání nejlepší kombinace hyperparametrů

V programu se u statistických modelů nachází tlačítko "Find Best Config", jež slouží pro vyhledání nejlepšího hyperparametru, resp. kombinace hyperparametrů. Proces funguje tak, že se spustí $n/2$ paralelních procesů, kde n odpovídá počtu logických procesorů systému na němž program běží. Výlučný přístup ke zdrojům je zajištěn, neboť každý proces běží na vlastních datech. Jediná oblast kde bylo potřeba zařídit synchronizaci procesů je při vracení výsledků jednotlivých kombinací hyperparametrů zpátky do prvotní instance modelu. Toto je zajištěno pomocí funkcí `acquire()` a `release()` na objektu `Lock` z knihovny `multiprocessing`.

Podle druhu modelu se zároveň liší způsob, kterým se vyhodnocuje kvalita jednotlivých kombinací hyperparametrů. U modelu ARIMA rozhoduje metrika RMSE 2.6 natrénovaného modelu z predikce na testovacím datasetu. U modelů SARIMA a VAR rozhoduje Akaikeho informační kritéruim 2.4.5, kde se model pouze natrénuje. Pro informaci se také ukládá Bayesovské informační kritérium 2.4.6.

4.6 Modely s neuronovou sítí

Do programu byly implementovány celkem dva druhy modelů s neuronovou sítí. Prvním z nich je model s rekurentní neuronovou sítí 2.5.6 využívající vrstvy dlouhé krátkodobé paměti 2.5.6. Druhý model využívá konvoluční neuronovou síť 2.5.7. Nedá se říct, že by některý z nich byl lepší než druhý, oba modely mají své výhody a nevýhody.

Hlavním rozdílem mezi těmito dvěma je ten, že rekurentní model kvůli svým paměťovým vlastnostem dokáže zachytit dlouhodobé závislosti v datech a dokáže se učit z minulosti, což jej dobře předurčuje k řešení úloh, jako je předpovídání cen akcií nebo předpověď počasí. Tento model však má větší výpočetní náročnost oproti konvolučnímu modelu, čímž pádem se déle učí. Jednou z výhod konvolučního modelu je zase schopnost učit se složité vzorce v datech s větší efektivitou než rekurentní model. Nevýhodou může být větší náchylnost k přetrénování.

4.6.1 Normalizace dat

Normalizace dat je proces, který se v neuronových sítích používá ke standardizaci rozsahu vstupních proměnných tak, aby vstupy měli stejnou škálu a stejné rozložení. Normalizace je důležitá, protože neuronové sítě jsou citlivé na rozsah vstupů. Pokud by byl rozsah vstupů příliš velký, mohl by se učící algoritmus stát nestabilním, což by vedlo k nepřesným předpovědím. Normalizace dat zajistí, že všechny vstupy budou mít stejný rozsah, a tedy i stejný dopad na síť.

V programu používanou normalizací je škálování min-max. Jedná se o změnu měřítka dat tak, aby všechny hodnoty spadaly do daného rozsahu, v tomto případě nastaveného na rozsah 0 až 1. Normalizace je důležitá také proto, že velikost vah v neuronové síti může ovlivnit přesnost modelu. Pokud jsou váhy příliš velké, může model příliš přizpůsobit data, což vede ke špatné generalizaci. Normalizace pomáhá zajistit, že váhy zůstanou malé, a proto je pravděpodobnější, že model bude dobře zobecňovat.

4.6.2 Učení s učitelem

Pro konverzi dat na problém nazývaný jako učení s učitelem 2.5.5 se v programu využívá objekt `TimeSeriesGenerator`. Jedná se o speciální typ generátoru sekvencí, který vezme posloupnost hodnot a převede je na sekvenci vstupů a výstupů. Vstupem je kombinace aktuálních nebo minulých hodnot časové řady, zatímco výstupem je další hodnota v posloupnosti řady. Model tak může využít minulé hodnoty časové řady k předpovědi následující hodnoty.

`TimeSeriesGenerator` umožňuje přizpůsobit délku vstupní posloupnosti a předpověď výstupu. To umožňuje modelu naučit se z dat časové řady složitější vzory. Pokud je například použita delší vstupní posloupnost, může se model z dat naučit složitější vzory, jako například sezónnost. Délku vstupní posloupnosti si určuje uživatel nastavením požadovaného počtu minulých hodnot při konfiguraci modelu, délka výstupní posloupnosti je pevně nastavena na 1 hodnotu.

4.6.3 Model s rekurentní neuronovou sítí

První vrstvou tohoto modelu je vrstva LSTM. Tato vrstva je určena k zachycení dlouhodobých závislostí v datech časových řad a také krátkodobých korelací. Vstupem jsou data časových řad ve tvaru (počet zpožděných hodnot, počet vstupních řad).

Tato vrstva má 100 neuronů a aktivační funkci "relu". Aktivační funkce "relu"je nelineární aktivační funkce, která zde slouží k zavedení nelinearity do modelu. Argument 'return_sequences' je nastaven na hodnotu True, takže výstupem bude posloupnost vektorů, nikoli jeden vektor. To umožňuje modelu zachytit časové závislosti ve více časových krocích. Zároveň je toto nastavení nutné z toho důvodu, že následná LSTM vrstva vyžaduje posloupnost vektorů, nikoli pouze jeden vektor.

Druhou vrstvou modelu je vrstva Dropout, která slouží k zabránění nadměrnému přizpůsobení modelu tím, že během trénování náhodně vynechává neurony. Tato vrstva má míru vynechání nastavenou na 0,4. To znamená, že 40 % neuronů bude během trénování náhodně vynecháno. Dropout byla přidána, neboť při experimentování byla zjištěna nestabilita modelu při predikcích na testovacích datech. Výstupem byla vyšší hodnota, a protože se hodnota používala pro předpověď další hodnoty, vedlo to k predikci ještě vyšší hodnoty. Nakonec byly predikované hodnoty tak velké, že model selhal. Tento problém se podařilo vyřešit mimo jiné právě přidáním vrstvy Dropout.

Po Dropout následuje další vrstva LSTM, která je podobně jako ta první určena k zachycení časových závislostí v datech, ale tato vrstva má 50 neuronů, což znamená, že by měla být schopna zachytit jiné druhy závislostí než ta první. Aktivační funkce této vrstvy je rovněž "relu".

Jako poslední je vrstva Dense, která slouží k mapování výstupu druhé vrstvy LSTM na finální výstup modelu. Počet neuronů v této vrstvě je roven počtu vstupních časových řad, což umožňuje modelu mapovat výstup vrstvy LSTM na požadovaný výstup.

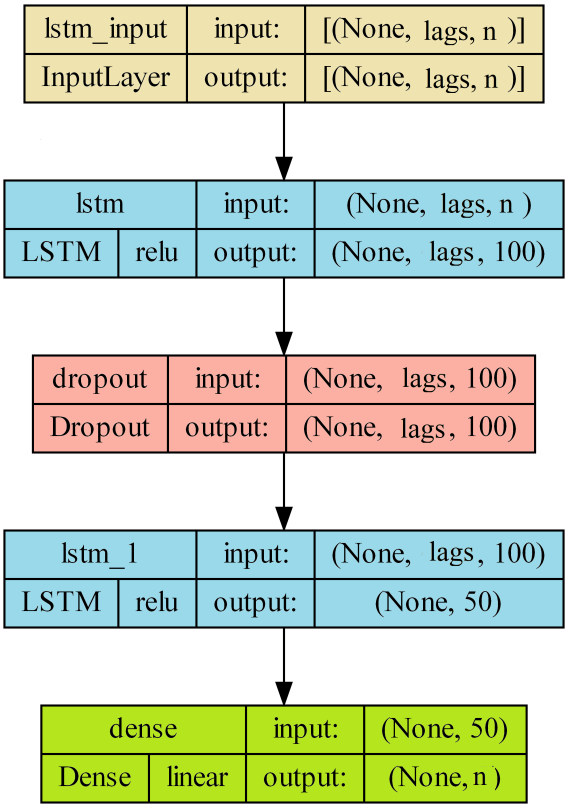
Nakonec je model sestaven pomocí optimalizátoru "adam"a ztrátové funkce "mse". Optimalizátor "adam"se používá k aktualizaci vah modelu během trénování a střední čtvercová chyba, která je nastavená jako ztrátová funkce se používá k výpočtu rozdílu mezi předpovídanými a skutečnými hodnotami.

Grafické znázornění jednotlivých vrstev tak jak jdou za sebou je na obrázku 4.3.

4.6.4 Model s konvoluční neuronovou sítí

První vrstvou modelu s konvoluční neuronovou sítí je vrstva Conv1D. Tvar vstupu je totožný jako u rekurentního modelu. Tento typ vrstvy se používá k detekci vzorů v datech časové řady provedením operace konvoluce na vstupu. Jejím účelem je efektivně extrahovat smysluplné vzory ze vstupních dat. Operace konvoluce je založena na kernelu (konvolučního jádra), což je sada vah, které se používají k provedení operace konvoluce. Velikost konvolučního jádra je zde nastavena na 3. To znamená, že konvoluční operace budou prováděny vždy ve 3 časových krocích. Počet filtrů je nastaven na 64. V každém časovém kroku tedy bude provedeno 64 různých konvolučních operací. Použitá aktivační funkce je "relu".

Další je vrstva Dropout, jež funguje obdobně jako u rekurentního modelu. Poté následuje další vrstva Conv1D se stejnými parametry jako první vrstva. Tato vrstva slouží k dalšímu získávání smysluplných vzorů z dat. Použitím dvou konvolučních vrstev lze z dat získat složitější vzory. První vrstva slouží k detekci jednoduchých vzorů v datech a druhá vrstva slouží k detekci složitějších vzorů. To pomáhá zlepšit přesnost a schopnost zobecnění modelu.



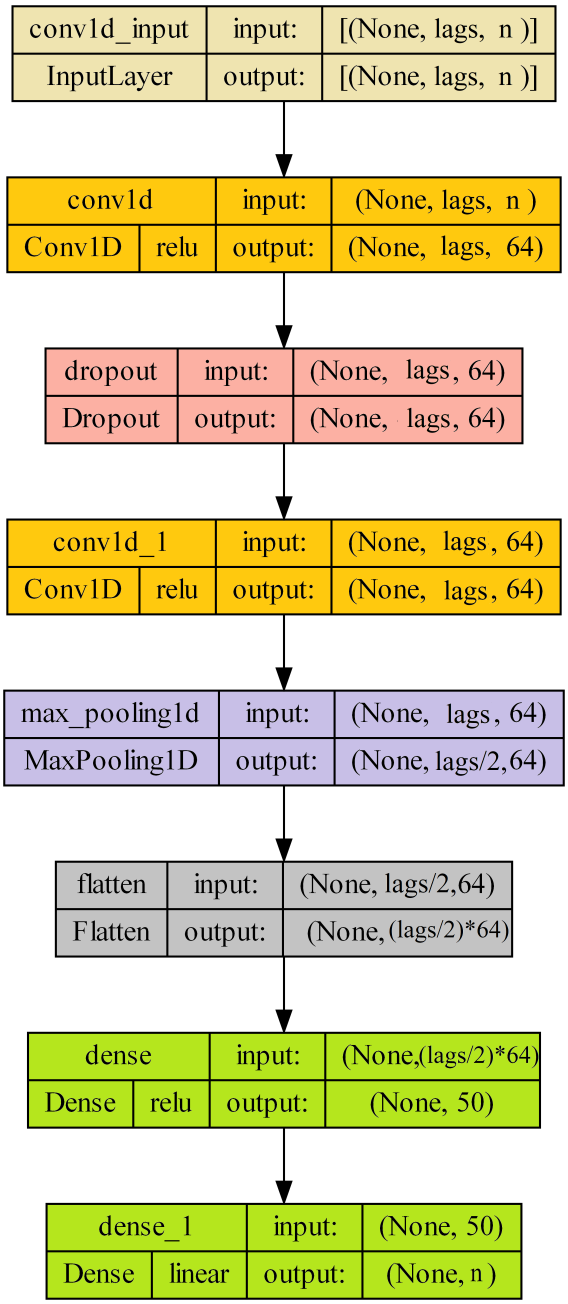
Obrázek 4.3: Schéma vrstev implementované RNN, proměnná lags odpovídá počtu minulých hodnot, proměnná n počtu vstupních časových řad.

Čtvrtou vrstvou je vrstva MaxPooling1D, jež se používá ke zmenšení velikosti vstupu kombinací hodnot ze sousedních časových kroků a výběrem maximální hodnoty každé skupiny. To pomáhá snížit počet parametrů v modelu a tím zlepšit jeho výkonnost. V tomto modelu je velikost skupiny nastavena na 2 a maximální hodnota bude tedy vzata ze skupin dvou časových kroků ((n-1, n), (n, n+1)).

Pátou vrstvou je vrstva Flatten, která v tomto modelu odpovídá za převod 3D výstupu z konvolučních vrstev na 1D pole. To je nezbytné pro vstup do následných vrstev Dense, které jako vstup vyžadují 1D pole. Vrstva Flatten převezme výstup konvolučních vrstev a "zploští"jej tak, aby bylo možné různé mapy příznaků spojit do jediného vektoru. Tento vektor pak může být použit jako vstup do vrstvy Dense.

Poslední dvě vrstvy jsou Dense. Předposlední z nich se používá k propojení výstupu předchozí vrstvy s výstupní vrstvou. Počet neuronů v této vrstvě je nastaven na 50 a použitá aktivační funkce je "relu". Poslední Dense má stejný počtem neuronů, jako je počet vstupních časových řad a slouží k vytváření předpovědí na základě výstupu předchozí vrstvy. Použitá aktivační funkce je lineární, což znamená, že výstup této vrstvy bude lineární kombinací vstupů.

Architektura modelu je znázorněna na obrázku 4.4.



Obrázek 4.4: Schéma vrstev implementované CNN, proměnná lags odpovídá počtu minulých hodnot, proměnná n počtu vstupních časových řad.

Kapitola 5

Testování

Tato kapitola pojednává o výsledcích testování kvality jednotlivých implementovaných modelů a jejich vyhodnocování na různých typech časových řad. Jednotlivé modely jsou otestovány na časových řadách různých složitostí. Testy se skládájí z výstupního grafu, kde jsou vidět původní data časové řady rozdělená na trénovací a testovací část, a pro každou tuto část je provedena predikce. Pod výstupním grafem následuje tabulka chybových metrik rozdělená opět na trénovací a testovací část.

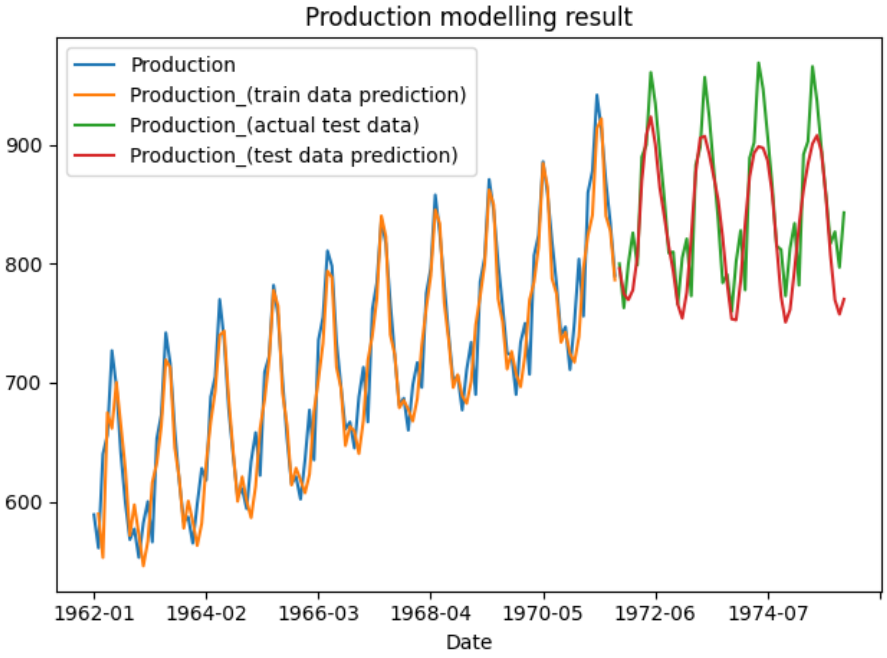
5.1 Testování statistických modelů na jednoduché časové řadě

Testování modelů ARIMA a SARIMA proběhne na časové řadě, která obsahuje trendovou a sezonní složku. Poměr rozdělení trénovací a testovací části je 70:30. Vstupní hyperparametry obou modelů byly určeny pomocí funkce "Find Best Config", více popsané v 4.5.2. Graf 5.1 zobrazuje výstup modelu ARIMA s hyperparametry $(p = 6, d = 1, q = 2)$.

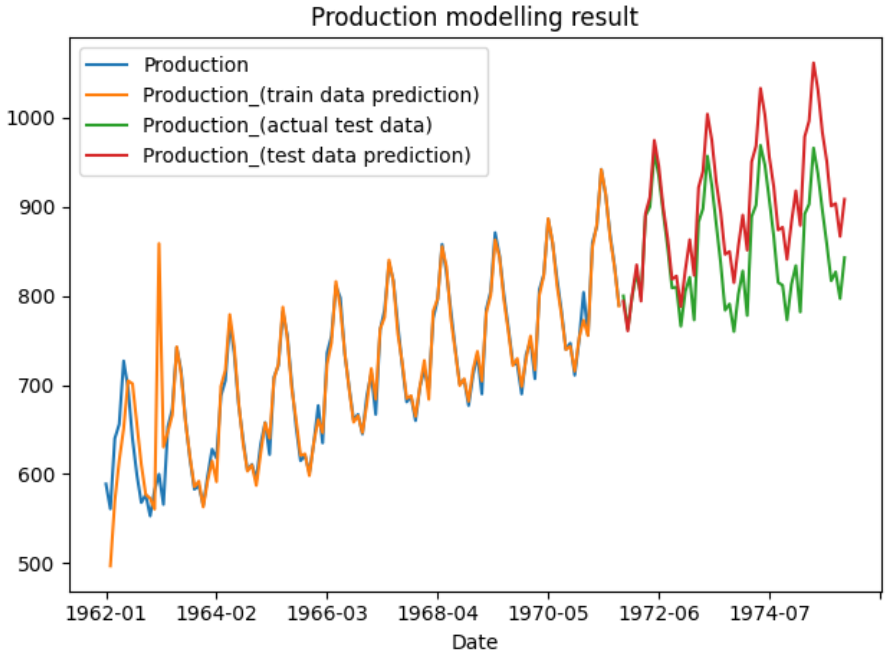
Druhý výstupní graf 5.2 reprezentuje model SARIMA s hyperparametry $(p = 1, d = 1, q = 3)(P = 3, D = 0, Q = 2, m = 6)$. Počáteční nestabilita tohoto modelu na trénovacích datech je způsobena nastavením hyperparametru m, jenž určuje periodicitu jedné sezóny. Tabulky 5.1 a 5.2 ukazují chybové metriky k jednotlivým výstupům.

Metrika	Hodnoty na trénovacích datech	Hodnoty na testovacích datech
MSE	705,791	1200,819
RMSE	26,567	34,653
MSLE	0,002	0,002
MAE	14,723	29,025
MAPE	2,9 %	3,3 %

Tabulka 5.1: Chybové metriky modelu ARIMA



Obrázek 5.1: Výstupní graf pro vyhodnocení modelu ARIMA.



Obrázek 5.2: Výstupní graf pro vyhodnocení modelu SARIMA.

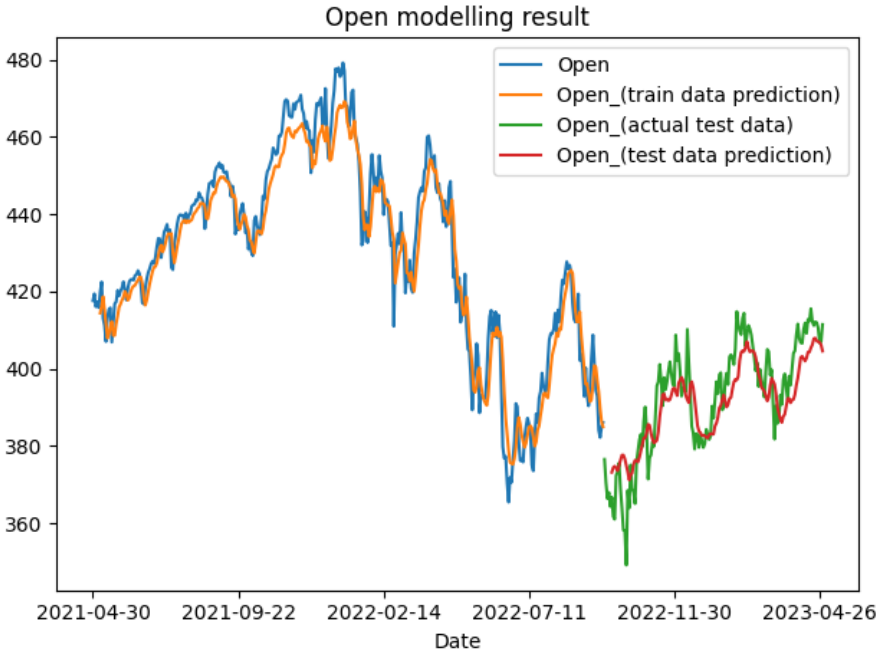
Metrika	Hodnoty na trénovacích datech	Hodnoty na testovacích datech
MSE	890,513	3324,453
RMSE	28,841	57,658
MSLE	0,002	0,004
MAE	4,784	55,719
MAPE	1,8 %	5,8 %

Tabulka 5.2: Chybové metriky modelu SARIMA

5.2 Testování modelů na složité časové řadě

Testování modelů s neuronovou sítí proběhne na řadě, reprezentující cenový vývoj akciového indexu S&P 500 za poslední dva roky. Model s rekurentní neuronocu sítí má nastaven parametr pro počet zpožděných hodnot na 5 a počet učicích epoch na 30. Výstup je na obrázku 5.3. Následuje výstup modelu 5.4 s konvoluční neuronovou sítí, který má nastaven počet zpožděných hodnot na 5 a počet epoch 15.

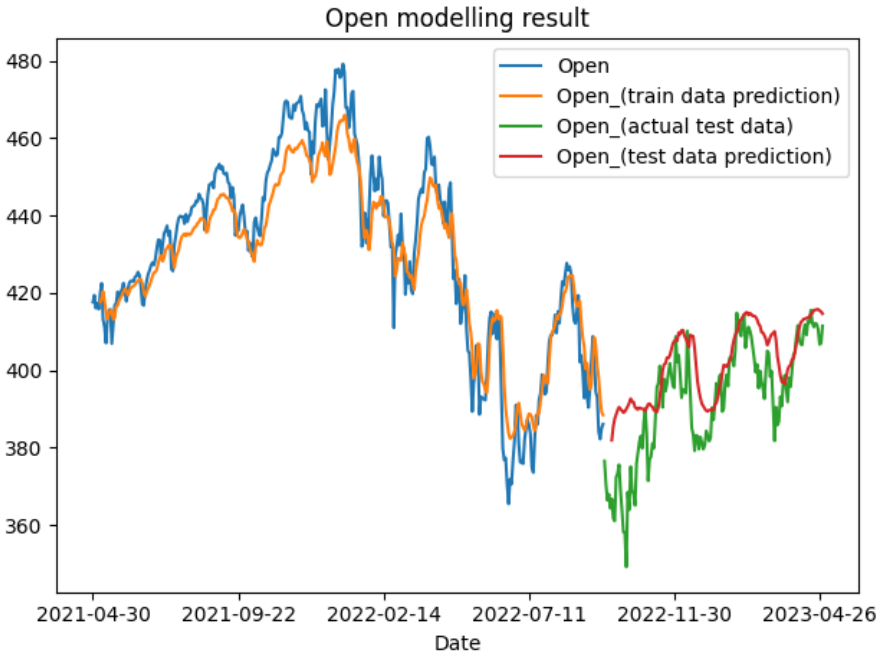
Na první pohled je vidět lepší přesnost modelu RNN. CNN má sice nastaven menší počet epoch, nicméně experimenty bylo zjištěno, že přesnost se zde s rostucím počtem epoch spíše snižuje. To může být způsobeno větší náchylností konvoluční neuronové sítě k přetrénování.



Obrázek 5.3: Výstupní graf pro vyhodnocení modelu RNN s parametry lags=5, epochs=30

Metrika	Hodnoty na trénovacích datech	Hodnoty na testovacích datech
MSE	41,822	62,617
RMSE	6,467	7,913
MSLE	0	0
MAE	4,459	5,274
MAPE	1,2 %	1,6 %

Tabulka 5.3: Chybové metriky modelu s rekurentní neuronovou sítí



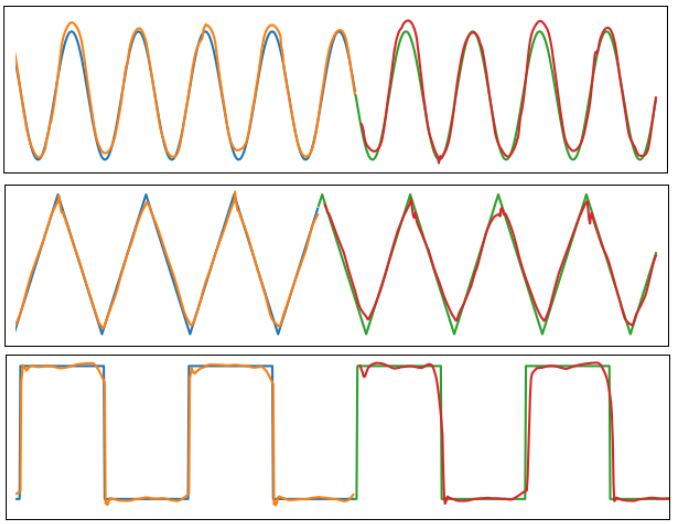
Obrázek 5.4: Výstupní graf pro vyhodnocení modelu CNN s parametry lags=5, epochs=15

Metrika	Hodnoty na trénovacích datech	Hodnoty na testovacích datech
MSE	61,828	151,156
RMSE	7,863	12,295
MSLE	0	0,001
MAE	5,576	6,916
MAPE	1,5 %	2,4 %

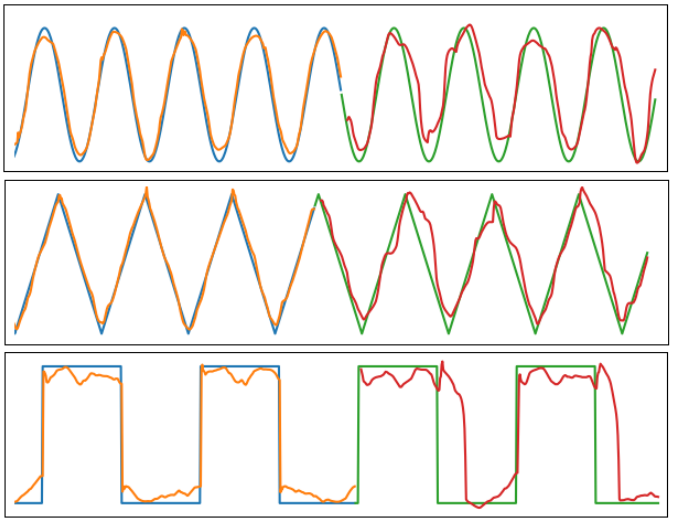
Tabulka 5.4: Chybové metriky modelu s konvoluční neuronovou sítí

5.3 Testování modelů na multivarietní predikci

Testování modelů s neuronovou sítí na multivarietní časové řadě, kde první řada reprezentuje funkci sinus, druhá řada má trojúhelníkový průběh a poslední řada má obdélníkový průběh. Výstupní grafy pro modely RNN 5.5 a CNN 5.6 jsou kvůli lepší čitelnosti přiblížené a bez legendy a os. Jednotlivé barvy čar nicméně odpovídají barvám čar předchozích výstupů z této kapitoly. Ačkoliv oba modely používají stejný počet zpožděných hodnot a stejný počet trénovacích epoch, je na první pohled patrné, že rekurentní model dosáhl lepších výsledků.



Obrázek 5.5: Výstupní graf pro multivarietní vyhodnocení modelu RNN s parametry lags=7, epochs=25



Obrázek 5.6: Výstupní graf pro multivarietní vyhodnocení modelu CNN s parametry lags=7, epochs=25

Kapitola 6

Závěr

Cílem práce bylo vytvořit program, který zvládne načíst data časových řad a bude umožňovat jejich predikci ve zvolené časové vzdálenosti. Tento cíl se podařilo naplnit. Byl navržen a vytvořen program s grafickým uživatelským rozhraním a důrazem na snadnou rozšiřitelnost, který umožňuje nejen predikovat, ale také vyhodnocovat kvalitu predikce časových řad jak pomocí statistických modelů, tak i modelů s neuronovou sítí. Modely si zároveň uživatel může nakonfigurovat. Jako bonus byly implementovány funkcionality pro paralelizované vyhledávání co možná nejlepší kombinace hyperparametrů u statistckých modelů a možnost ukládání natrénovaných nebo nakonfigurovaných modelů.

Je možné modelovat univarierní i multivarietní časové řady. V prvním případě se modeluje jedna řada, v druhém případě je paralelně modelováno více řad najednou. Zde se nabízí prostor pro možné rozšíření kdy by bylo například umožněno modelovat časovou řadu z jiných řad, nebo třeba předpovídat vývoj dvou výstupních řad ze třech či více vstupních řad. Dalším vylepšením by mohlo být umožnění vícekrokové predikce u neuronových modelů tak, aby bylo možné v jednom iteračním kroku předpovědět více než jednu výstupní hodnotu.

Literatura

[1] Arbel, N. *How LSTM networks solve the problem of vanishing gradients* [online]. 2018 [cit. 2023-02-26]. Dostupné z: https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577.

[2] Brockwell, P. J. a Davis, R. A. *Introduction to Time Series and Forecasting, Second Edition.* Springer, 2002.

[3] Brownlee, J. *How to Use the Keras Functional API for Deep Learning* [online]. 2020 [cit. 2022-12-18]. Dostupné z: https://machinelearningmastery.com/keras.

[4] Burnham, K. P. a Anderson, D. R. *Model Selection and Multimodel Inference.* Springer New York, NY, 2002.

[5] Chatfield, C. *The Analysis of Time Series: An Introduction.* Chapman & Hall/CRC.

[6] Chatfield, C. Journal of Forecasting: Model Uncertainty and Forecast Accuracy. [online]. University of Bath, UK. 1996, s. 495 – 508, [cit. 2022-12-2]. DOI: 10.1002/(SICI)1099-131X(199612)15:7<495::AID-FOR640>3.0.CO;2-O.

[7] Cochrane, J. H. *Time Series for Macroeconomics and Finance.* Graduate School of Business, University of Chicago, 1997.

[8] Dotis, A. *Autocorrelation in Time Series* [online]. 2019 [cit. 2022-12-02]. Dostupné z: https://dganais.medium.com/autocorrelation-in-time-series-c870e87e8a65.

[9] Godwin, J. A. *Time Series Analysis* [online]. 2021 [cit. 2022-11-02]. Dostupné z: https://towardsdatascience.com/time-series-analysis-7138ec68754a.

[10] Guo, M. a Tseng, Y. K. A Comparison between Linear and Nonlinear Forecasts for Nonlinear AR Models. *Journal of Forecasting.* John Wiley & Sons. 1997, sv. 16, č. 7, s. 491–508, [cit. 2022-12-17]. ISSN 0277-6693.

[11] HIPEL, K. a McLEOD, A. *Time Series Modelling of Water Resources and Environmental Systems.* Elsevier, 1994.

[12] Kim, S. a Kim, H. A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting* [online]. Elsevier. 2016, sv. 32, č. 3, s. 669–679, [cit. 2022-12-15]. DOI: https://doi.org/10.1016/j.ijforecast.2015.12.003. ISSN 0169-2070. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0169207016000121.

[13] Kok Joost N., R. G. *Handbook of Natural Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 9783540929093.

[14] Mushailov, J. *LSTM Framework For Univariate Time-Series Prediction* [online]. 2021 [cit. 2023-02-26]. Dostupné z: https://towardsdatascience.com/lstm-framework-for-univariate-time-series-prediction-d9e7252699e.

[15] Pankratz, A. *Forecasting with Univariate Box - Jenkins Models: Concepts and Cases*. John Wiley&Sons, 1983.

[16] Prabhakaran, S. *Augmented Dickey Fuller Test (ADF Test) - Must Read Guide* [online]. 2019 [cit. 2023-01-05]. Dostupné z: https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/.

[17] Rink, K. *Time Series Forecast Error Metrics You Should Know* [online]. 2021 [cit. 2022-12-15]. Dostupné z: https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27#:~:text=The%20mean%20absolute%20percentage%20error,values%20divided%20by%20the%20actuals.

[18] Seabold, S. a Perktold, J. Statsmodels: Econometric and Statistical Modeling with Python. *Proceedings of the 9th Python in Science Conference*. Leden 2010, sv. 2010, s. 57–61.

[19] Sharma, S. *Activation Functions in Neural Networks* [online]. 2017 [cit. 2023-02-05]. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[20] Soni, D. *Supervised vs. Unsupervised Learning* [online]. 2018 [cit. 2023-02-09]. Dostupné z: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[21] Versloot, C. *Relu-sigmoid-and-tanh-todays-most-used-activation-functions* [online]. 2022 [cit. 2023-01-23]. Dostupné z: https://github.com/christianversloot/machine-learning-articles/blob/main/relu-sigmoid-and-tanh-todays-most-used-activation-functions.md.

[22] Zivot, E. a Wang, J. *Modeling Financial Time Series with S-PLUS*. Second Edition. New York, NY: Springer New York. ISBN 978-0-387-27965-7.

Příloha A

Obsah přiloženého paměťového média

/doc/xdvora3d_casove_rady.pdf – Technická zpráva ve formátu pdf.

/doc/src – Soubory pro vytvoření technické zprávy.

/BP/src – Zdrojové kódy vztahující se k vytvořenému programu.

/BP/ – Ostatní soubory a složky vztahující se k vytvořenému programu.