VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KLASIFIKÁTOR STAVŮ DOPRAVY A DETEKTORŮ

CLASSIFIER OF TRAFFIC STATES AND DETECTORS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Ondřej Březina

AUTHOR

VEDOUCÍ PRÁCE — Ing. Ilona Janáková, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Ondřej Březina ***ID:*** 240318

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Klasifikátor stavů dopravy a detektorů

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout klasifikátor pro posouzení stavu dopravně-technologických informací z inteligentních dopravních systémů (ITS) – časové řady počtu projetých vozidel a průměrných rychlostí (případně dalších měřených parametrů). Předpokládá se klasifikace podle stavu dopravy – např. volný tok, špička, nehoda atd. a podle stavu detektorů – krátkodobý výpadek, detektor mimo provoz, různé chyby detektoru, výpadek lokality apod.

1. Seznamte se s danou problematikou.
2. Proveďte rešerši existujících přístupů.
3. Navrhněte vhodné přístupy ke klasifikaci.
4. Vybrané postupy implementujte.
5. Vše řádně otestujte na dodaném vzorku reálných dat, případně jednotlivé stavy vhodně simulujte.
6. Získané výsledky zhodnoťte. Definujte omezující podmínky.

DOPORUČENÁ LITERATURA:

JAGADEESH, George R., George R. DHINESH a Thambipillai SRIKANTHAN. Method for accuracy assessment of aggregated freeway traffic data. IET Intelligent Transport Systems [online]. 2014, 8(4), 407-414 [cit. 2023-08-17]. ISSN 1751-9578. Dostupné z: doi:10.1049/iet-its.2013.0094

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 22.5.2024

Vedoucí práce: Ing. Ilona Janáková, Ph.D.

Ing. Miroslav Jirgl, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce se zabývá popisem zdrojů dopravních dat a metodami analýzy těchto dat. Dále jsou v práci navrženy a implementovány stavy pro klasifikaci dopravy a detektorů v prostředí MATLAB. Na závěr jsou vymodelovány modely pro dopravní tok, které lze použít i pro predikci dopravního toku.

KLÍČOVÁ SLOVA

Inteligentní dopravní systém, dopravní detektory, časová řada, SARIMA, Box-Jenkins, predikce časové řady, MATLAB

ABSTRACT

The work deals with the description of sources of traffic data and methods of analysis of these data. Furthermore, states for the classification of traffic and detectors in the MATLAB environment are designed and implemented in the work. Finally, traffic flow models are modeled, which can also be used for traffic flow prediction.

KEYWORDS

Intelligent transport system, traffic detectors, time series, SARIMA, Box-Jenkins, time series prediction, MATLAB

BŘEZINA, Ondřej. *Klasifikátor stavů dopravy a detektorů*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2024. Vedoucí práce: Ing. Ilona Janáková, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Ondřej Březina
VUT ID autora:	240318
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Klasifikátor stavů dopravy a detektorů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce paní Ing. Iloně Janákové, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V dnešní době stále narůstá počet vozidel na silnicích a ve městech. Tento jev vede často k dopravním zácpám, které zpomalují provoz a znečišťují ovzduší. Inteligentní dopravní systémy (ITS) představují efektivní způsoby řízení a optimalizace dopravy.

Hlavními cílemi ITS jsou zvýšit bezpečnost, zajistit plynulost provozu a tím snížit dopravní zácpy a minimalizovat dopady na životní prostředí. Využívají pro to mnoho senzorů a technologií. Mezi ně se řadí systémy pro řízení křižovatek, dynamické informační tabule, různé detektory rychlosti, detektory nehod, elektronické výběry mýtného, navigační systémy a další.

V této práci se zaměřím na systémy sběru a metody analýz dopravních dat a pokusím se navrhnout klasifikační nástroje pro detekci výpadků měřicích stanic, detekci kongescí (dopravních zácp) a pokusím se namodelovat dopravní tok.

1 Zdroje dat

Tato kapitola se zabývá přehledem hlavních zdrojů dat v dopravě. Mezi hlavní zástupce se řadí indukční smyčky, různé typy kamerových systémů a data z plovoucích vozidel.

1.1 Dopravní veličiny

Před popisem zdrojů dat, je nutné zmínit, jaké dopravní veličiny se vyskytují a měří. [1]

1.1.1 Rychlost

Rychlost vozidel dělíme na **okamžitou rychlost** a **průměrnou rychlost**. Okamžitá rychlost je měřena v konkrétním místě (bodě) na vozovce. Průměrná rychlost označuje rychlost vozidel za časový úsek a měří se obvykle na určitém úseku vozovky.

1.1.2 Dojezdová doba

Další veličinou, která se v dopravě měří je doba dojezdu. To je doba, za kterou se vozidlo přesune z místa A do místa B.

1.1.3 Dopravní intenzita

Dopravní intenzita vyjadřuje počet vozidel za jednotku času. Značí se q a má jednotku *počet vozidel/hod.*

$$q = \frac{N}{T} \tag{1.1}$$

, kde N je počet vozidel a T je časová perioda.

1.1.4 Dopravní hustota

Další veličinou je dopravní hustota. Vyjadřuje počet vozidel na určitém úseku vozovky. Značí se k a nejčastěji má jednotku *voz./km.* Lze vypočítat podle vztahu 1.2, kde q je intenzita a u je rychlost.

$$k = \frac{q}{u} \tag{1.2}$$

1.2 Indukční smyčky

Indukční smyčky jsou oblíbenou metodou detekce vozidel. Fungují na principu vzájemné indukčnosti vozidla s cívkou (smyčkou). Vozidlo ovlivňuje magnetické pole

cívky, což způsobí změnu frekvence obvodu nebo fázový posun. Touto interakcí vozidla s cívkou vzniká tzv. magnetický profil, který je časově závislý a ovlivněný typem vozidla, vzdáleností náprav a dalšími parametry vozidla. Magnetický profil umožňuje měření rychlosti a rozpoznání typu vozidla, jestli se jedná např. o osobní nebo nákladní vozidlo. [2] Výkonnost smyček klesá v průběhu používání. Je to dáno povětrnostními podmínkami, opotřebením vozovky, nesprávnou instalací nebo průjezdem přetížených vozidel [3].

Hlavní nevýhodou smyček je, že se jedná o intrizuvní metodu měření. Smyčky musíme zařezat (zabudovat) do vozovky.

Existují dva způsoby zabudování smyček do vozovky, jednosmyčkové a dvousmyčkové.

Jednosmyčkové zapojení je založené na měření doby průjezdu vozidla přes smyčku. Ze znalosti délky vozidla, délky smyčky a času průjezdu můžeme vypočítat rychlost vozidla podle vzorce 1.3. Hlavní nevýhodou tohoto zapojení je, že neznáme délku projíždějícího vozidla.

$$v = \frac{l}{t_{off} - t_{on}} \tag{1.3}$$

, kde l je délka vozidla + délka smyčky, t_{on} je čas sepnutí smyčky a t_{off} je čas vypnutí.

Z tohoto důvodu se častěji používá **dvousmyčkové řešení**. Jedná se o dvě indukční smyčky zapojeny za sebou. Známe tedy přesnou vzdálenost mezi smyčkami a na základě rozdílu času sepnutí jednotlivých smyček můžeme spočítat rychlost projíždějícího vozidla podle vzorce 1.4, kde Δt je rozdíl času zapnutí smyček. Zásadní nevýhodou tohoto zapojení je, nachází-li se mezi smyčkami více vozidel, tak dochází ke spuštění první smyčky ještě dříve, než bylo změřeno první vozidlo. Dochází tedy k nepřesné kalkulaci rychlosti.

$$v = \frac{s}{\Delta t} \tag{1.4}$$

1.3 Kamerové systémy

Měření parametrů dopravy pomocí kamerových systémů se stává stále více populární. Jejich hlavní výhodou je, že se jedná o neintruzivní systémy. Systémy, které nevyžadují zásah do struktury vozovky, který je potřebný např. u indukčních smyček. Mezi nevýhody patří jejich přesnost. Ta je ovlivněna změnou osvětlení, stínů, různými odrazy, počasím a nestálostí kamery (camera jitter) [3].

Existují různé typy kamerových systémů, které se můžou lišit typem informací které poskytují. Jedná se např. o měření úsekové rychlosti (MUR), křižovatkové

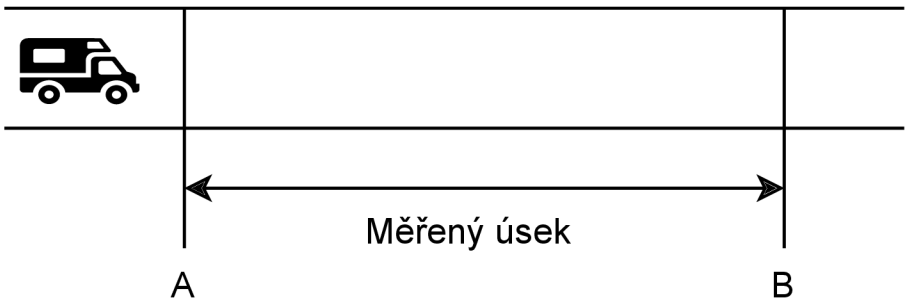
detektory, strategické detektory nebo komplexní telematický a dohledový systém (KTDS).

1.3.1 Měření úsekové rychlosti

Měření úsekové rychlosti pomocí kamer zjišťuje průměrnou rychlost projíždějícího vozidla na předem stanoveném úseku.

Principem měření je detekce času, který vozidlo potřebuje na projetí stanoveného úseku.

Na obrázku 1.1 je schéma MUR. V místě A je zaznamenán čas vjezdu vozidla do úsekového měření. V místě B je detekován čas výjezdu vozidla. Vozidlo je při vjezdu a výjezdu identifikováno pomocí registrační značky. Po výjezdu z měřeného úseku je spočítána jeho průměrná rychlost pomocí vztahu pro výpočet rychlosti 1.4, kde v je výsledná průměrná rychlost, s je délka měřeného úseku a t je čas potřebný k projetí úseku, respektive rozdíl času v místech A,B.



Obr. 1.1: Schéma MUR

1.3.2 Ostatní kamerové detekční systémy

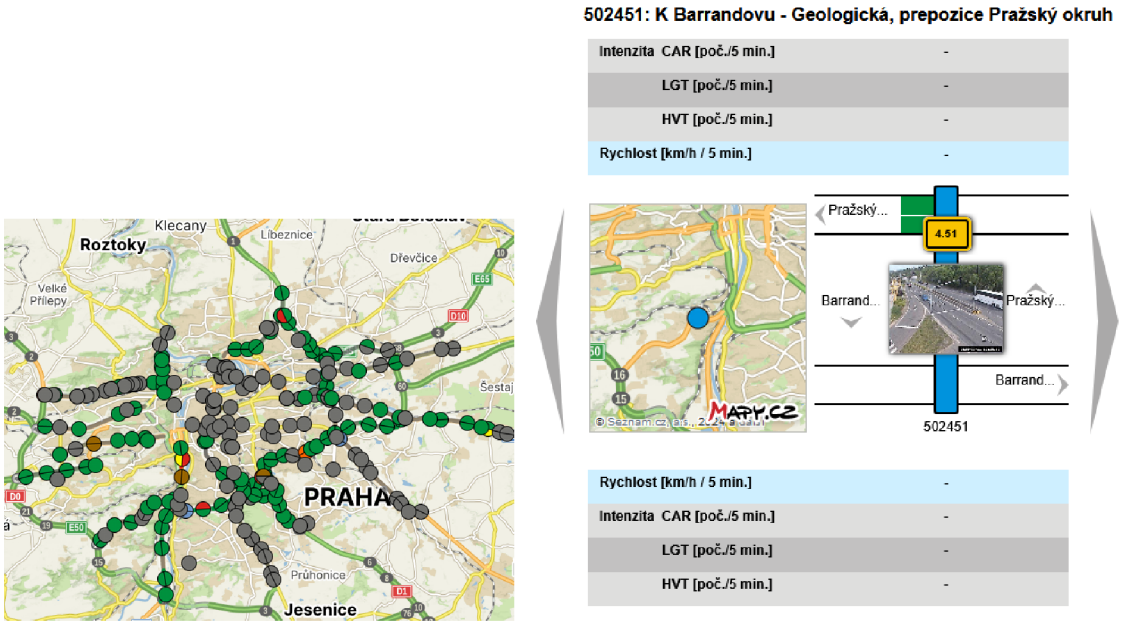
Jedná se o již zmíněné křižovatkové detektory, strategické detektory nebo komplexní telematické a dohledové systémy.

Křižovatkové detektory poskytují informace o aktuálních stavech v křižovatce. Mohou detekovat např. průjezdy na červenou nebo počty vozidel stojích na křižovatce.

Strategické detektory jsou umístěny minimálně 100 metrů před křižovatkou a poskytují např. informace o intenzitě a rychlosti přijíždějících vozidlech ke křižovatce. Tyto informace slouží k dynamickému řízení dopravy.

Komplexní systémy se skládají z mnoha video detektorů s přídavnými senzory poskytující přenos obrazu, dat a dopravních telematických informací. Slouží k zajištění telematických funkcí a operátorský dohled nad dopravou ve městě. Na obrázku

1.2 vlevo vidíme rozmístění kamer a senzorů patřících do systému KTDS v Praze. Na stejném obrázku vpravo je ukázka rozhraní takového systému. Vidíme, že tento senzor poskytuje informace o rychlosti, počtu vozidel a jejich kategoriích v 5 minutových intervalech.



Obr. 1.2: Ukázka KTDS (vlevo) a rozhraní KTDS v Praze (vpravo)

1.4 Radary

Radary vysílají mikrovlnné signály s určitou frekvencí a využívají Dopplerův jev, pro měření rychlosti vozidla. Podstatou jevu je závislost změny frekvence vysílaného a přijatého signálu při pohybu vozidla směrem k vysílači nebo od něj.[4] Aby mohlo probíhat kontinuální měření, je signál modulován v čase. [5]

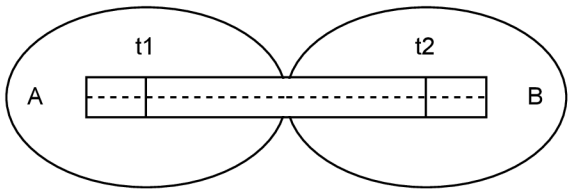
1.5 Data z plovoucích vozidel

Anglicky Floating Car Data (FCD). Jedná se o sběr polohových dat pomocí globálního navigačního systému nebo mobilních sítí, které jsou následně anonymizovány a analyzovány.

1.5.1 Mobilní sítě

V dnešní době komunikujeme přes mobilní telefony. Data o hovorech jsou uložena v databázích telefonních operátorů. V databázi je uložena identifikace telefonu, za-

čátek a konec hovoru a poloha vysílače, kde hovor začal a kde hovor skončil. Pokud je hovor kratší než 15 minut a počáteční vysílač je jiný než koncový, lze toho využít pro analýzu počtu a pohybu vozidel.[6] Pro ilustraci slouží obrázek 1.3.



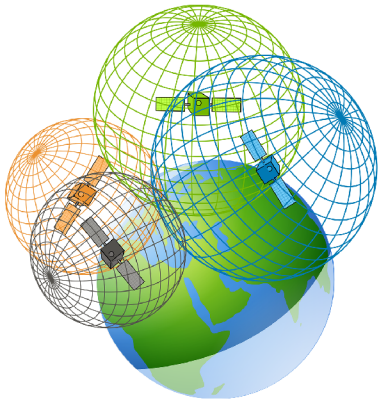
Obr. 1.3: Ilustrace sběru dat pomocí mobilní sítě

Hovor započne v čase t_1 v buňce A a skončí v čase t_2 v buňce B. Buňky vymezují dosah vysílače A, B.

1.5.2 Satelitní systémy

Určování polohy vozidel pomocí satelitních systémů je mnohem přesnější než využití mobilních sítí. Globální navigační satelitní systém (GNSS) umožňuje zařízením získat polohu na základě procesu zvaného trilaterace. Trilaterace je metoda určování polohy měřením vzdálenosti od minimálně tří referenčních bodů. V případě GNSS se měří doba letu signálu od satelitu k přijímači. Pro dosažení vyšší přesnosti se využívá signálů z mnoha satelitů. [7]

Na obrázku 1.4 je ukázka GNNS. Koule kolem satelitů jsou naměřené vzdálenosti a na jejich průsečíku je pak přesná poloha příjimače.[7]



Obr. 1.4: Ukázka GNSS, koule kolem satelitu představuje naměřenou vzdálenost [7]

Data FCD se sbírají obvykle z flotilových nebo firemních vozidel, např. taxíků. Jejich data o poloze a rychlosti jsou přenášena do informačního centra, kde jsou propojeny s mapou. A následnou analýzou se pak získávájí data o době dojezdu a průměrné rychlosti. [8]

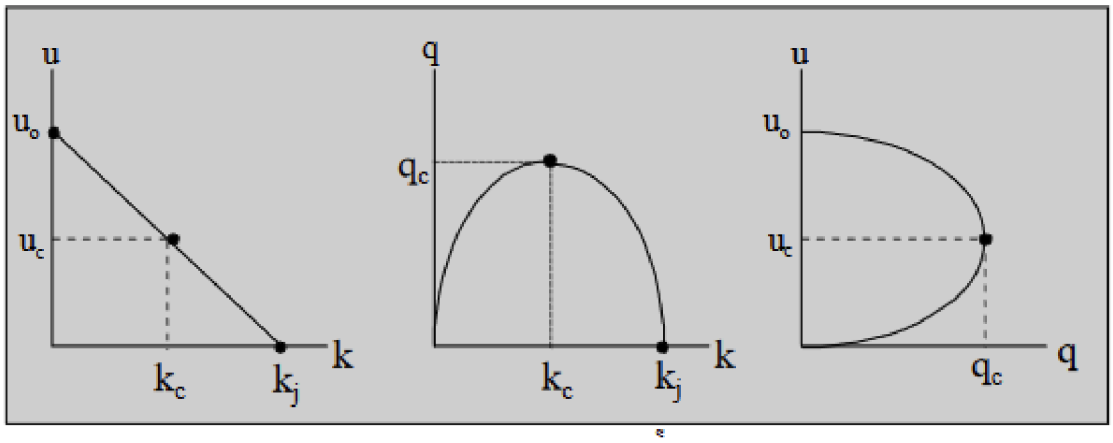
2 Metody analýzy dat

V této části práce jsou popsané nejpoužívanější metody zpracování dopravních dat. Nejzákladnější metodou je zachování konzistence dat. Dále jsou to metody na základě analýzy historických dat a pomocí neuronových sítí.

2.1 Konzistence

2.1.1 Fyzická konzistence

Jedná se o metody, které jsou založeny na kontrolách rozsahu očekávaných minimálních a maximálních hodnot jednoho či více parametru současně. Tyto metody jsou závislé na lokalitě umístění senzoru a limity parameterů se budou lišit. Např. pokud nejsou detekována vozidla, pak logicky nemůžeme spočítat jejich průměnou rychlost. Data jsou tedy chybná, pokud při nulovém počtu vozidel obsahují jejich průměrnou rychlost.

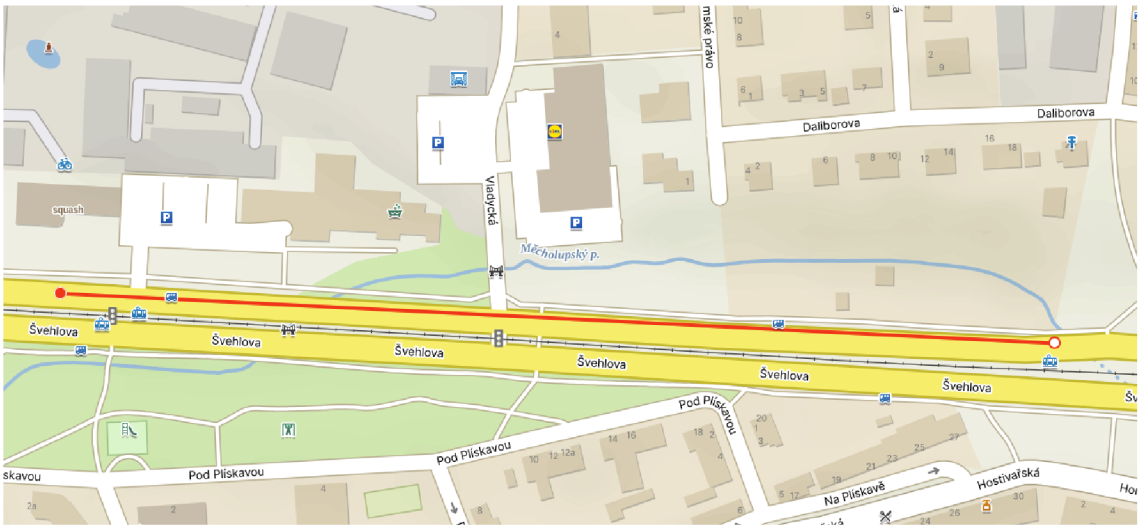


Obr. 2.1: Fundamentální grafy dopravy predikčních modelů [9]

„Graf vlevo na obrázku 2.1 znázorňuje vztah mezi rychlostí a hustotou dopravního proudu. Prostřední graf znázorňuje vztah mezi intenzitou a hustotou dopravního proudu. Poslední graf znázorňuje vztah mezi rychlostí a intenzitou. u_0 značí rychlost volného průjezdu, u_c je kritická rychlost, pod kterou začíná tvorba kongescí, k_c je kritická hustota, k_j je maximální možná hustota a q_c je kritická intenzita."[1]

2.1.2 Časoprostorová konzistence

Metoda je založena na principu korelace mezi více senzory. Korelace je termín označující míru vzájemné vazby mezi dvěmi veličinami. Pokud jsou dva senzory blízko sebe, např. senzor ve vedlejším pruhu nebo senzory na sebe navazující, časové řady



Obr. 2.2: Umístění MUR senzorů Švehlova I, II

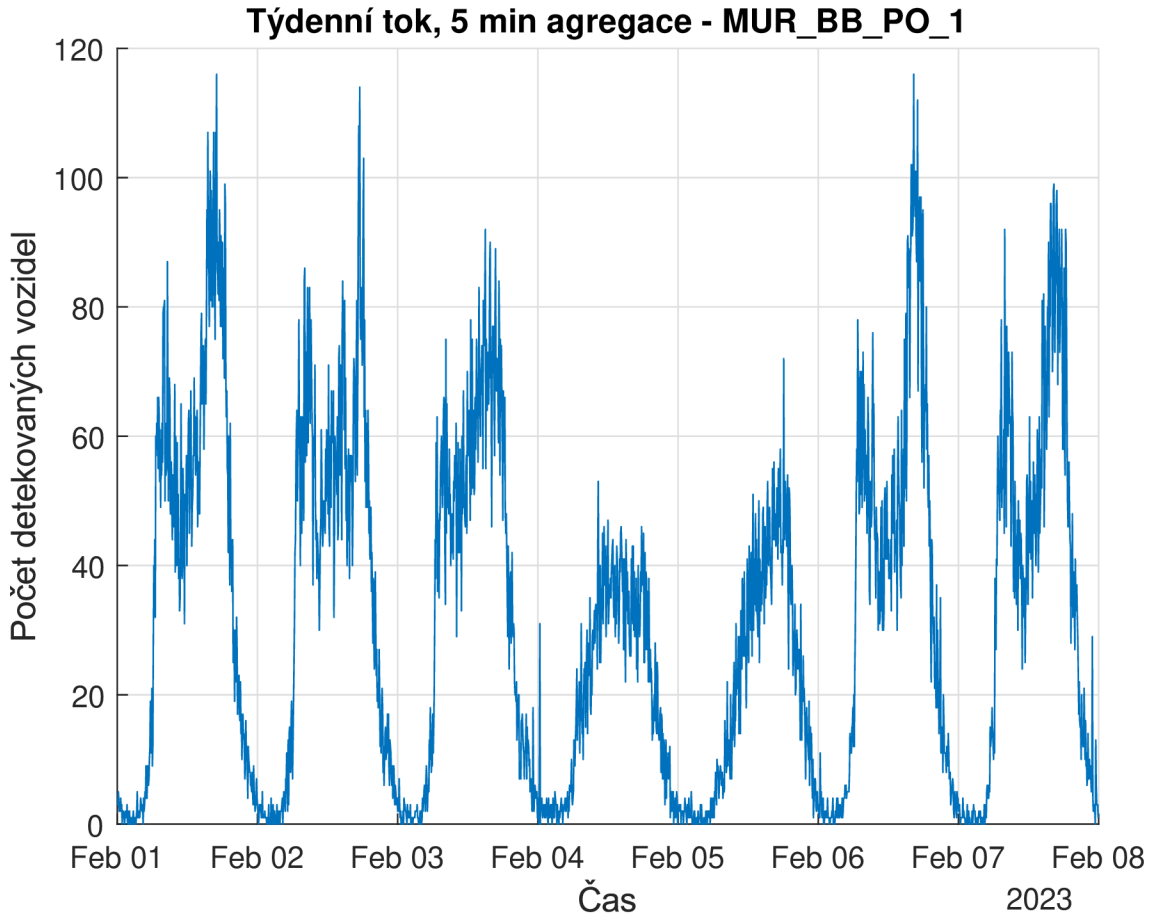
by měli být podobné Např. jakou jsou detektory na ulici Švehlova, viz obrázek 2.2, na kterém je zobrazeno umístění senzorů, vzdálené cca 430 metrů.

2.2 Analýza historických dat

Další skupinou metod jsou analýzy historických dat analýzou časové řady. Jedná se o dekompoziční metodu, autoregresní modely, modely s klouzavým průměrem, Kalmanův filtr a umělé neuronové sítě.

2.2.1 Dekompoziční metoda

Pokud se podíváme na graf týdenního toku vozidel, obrázek 2.3, lze si všimnout, že se denně opakuje jistý vzorec - v pracovní dny nastává ranní a odpolední špička a o víkendu (4. a 5. února) bez špiček. Tuto časovou řadu tedy můžeme rozdělit na trendovou, sezónní, cyklickou a náhodnou složku. [10] Trendová složka vyjadřuje, jestli bude časová řada stoupat nebo klesat. Sezónní složka vyjadřuje pravidelně se opakující odchylku od trendové složky. Cyklická složka představuje kolísání okolo trendu, které se však opakuje s různou periodou nebo amplitudou a je obtížně pozorovatelné. Náhodnou (reziduální) složkou jsou náhodné výkyvy v dopravě, např. nehoda. Tvoří ji také tzv. bílý šum - nekorelovaná náhodná hodnota s nulovou střední hodnotou. [11] Dekompozice se používá pro identifikaci anomálií [12].



Obr. 2.3: Ukázka týdenního toku vozidel

2.2.2 Autoregrese

Autoregresní model (AR) je definován lineární kombinací předešlých hodnot modelované proměnné y_t. AR řádu p označujeme $AR(p)$ a je definován rovnicí 2.1

$$y_t = \theta_0 + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \epsilon_t. \tag{2.1}$$

, kde θ_0 je konstanta φ_p jsou parametry AR modelu a y_{t-p} jsou hodnoty v čase $t - p$ a ϵ_t je náhodná složka (reziduum). Řád p je určen maximálním zpožděním. [11] Model AR bude stacionární, pokud budeme mít stacionární data nebo budou parametry AR $|\varphi_p| < 1$. [1]

Stacionární a nestacionární časová řada

Chování časové řady můžeme označit jako stacionární nebo nestacionární. Stacionaritu rozlišujeme jako striktní stacionaritu nebo slabou stacionaritu. Striktní stacionarita není závislá v čase. Slabá stacionarita označuje stabilní střední hodnotu,

konstantní rozptyl a časově invariantní řadu. Tedy řadu, která je závislá pouze na časovém rozdílu a ne na konkrétních časových okamžicích.

Pokud časová řada vykazuje např. trend nebo sezónní složku, pak o ní říkáme, že je nestacionární. [10]

Nestacionární časovou řadu můžeme převést na stacionární pomocí diference Δy.

$$\Delta y_t = y_t - y_{t-1} \tag{2.2}$$

Pokud řada bude stále nestacionární, diferenci opakujeme.

2.2.3 Klouzavý průměr

Proces klouzavých průměrů (MA) je definován lineární kombinací předešlých hodnot odchylky modelu od skutečné hodnoty. MA řádu q MA(q) je definován rovnicí 2.3.

$$y_t = \theta_0 + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \tag{2.3}$$

, kde ϵ_{t-q} jsou rezidua v kroku $t - q$ a θ_q jsou parametry procesu a řád q je dán zpožděním. [13]

2.2.4 Smíšený integrovaný proces

Smíšený integrovaný proces modelů AR, I, MA - ARIMA. I(d) je integrovaný model a vyjadřuje zde diferenciaci řady, aby se časová řada mohla stát stacionární. ARIMA model je určený pro časové řady se změnami úrovní a trendu (nestacionární časové řady). Zapisujeme jako ARIMA(p, d, q). Parametr p je řád autoregrese, d je počet diferencí a q určuje řád klouzavého průměru. [14]

$$y_t = \theta_0 + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \tag{2.4}$$

2.2.5 Sezónní integrovaný proces

Jedná se o sezónní integrovaný proces neboli SARIMA. Značíme $(S)ARIMA(p, d, q)(P, D, Q)_m$, kde (P, D, Q) jsou sezónní parametry a m je počet sezón. Sezónní parametry mají podobný význam jako parametry (p, d, q). Rozdíl je v tom, že při tvoření modelu se musíme posouvat o $(t - mD)$, zohledňujeme tedy hodnoty v předchozí sezóně. [11]

Pro představu. Vytváříme model $ARIMA(1, 0, 0)(0, 1, 1)_4$. Výsledný model bude vypadat následovně:

$$y_t = \theta_0 + \varphi_1 y_{t-1} + \epsilon_t + \theta_1\epsilon_{t-4\cdot1} \tag{2.5}$$

2.2.6 Box-jenkinsonova metodologie

Jedná se o postup pro sestavování modelů, kde tato metologie bere v úvahu korelované náhodné veličiny a využívá se zde korelační analýzy - analýza souvislostí mezi jednotlivými pozorováními. „Je mnohem flexibilnější než rozklad časové řady. Lépe se adaptuje na změny v řadě.“[11] Výstavbu modelu lze rozdělit do 3. částí. [15]

- Identifikace modelu
- Odhad parametrů modelu
- Ověřování modelu

Identifikace modelu spočívá ve výběru vhodného typu modelu na základě sestavení grafu, různých statistických testech nebo také z informace zda je řada stacionární nebo jestli řada vykazuje sezónnost.

„Základním prostředkem podávajícím informaci o charakteru stochastického procesu je autokorelační funkce (ACF) a parciální autokorelační funkce (PACF).“ [15]

Hodnota ACF ρ_k pro časovou řadu y_t je v bodě $k \in \mathbf{Z}$ rovna

$$\rho_k = \frac{\gamma_k}{\sigma_y^2} \tag{2.6}$$

, kde $|\rho_k| \leq 1$ a γ_k je autokovarianční funkce, jejíž hodnota je

$$\gamma_k = cov(y_t, y_{t+k}) = E(y_t - m)(y_{t+k} - m) \tag{2.7}$$

, kde $m = E(y_t)$ je střední hodnota, σ_y^2 je rozptyl dané stacionární řady.

PACF vyjádřuje závislost mezi y_t a y_{t+k} při odstranění vlivu $y_{t+1}, y_{t+2}, \ldots, y_{t+k-1}$.

$$PACF = cor(y_t, y_{t+k} | y_{t+1}, y_{t+2}, \ldots, y_{t+k-1}) \tag{2.8}$$

ACF a PACF lze graficky vyjádřit korelogramem, viz Obrázek 2.4.

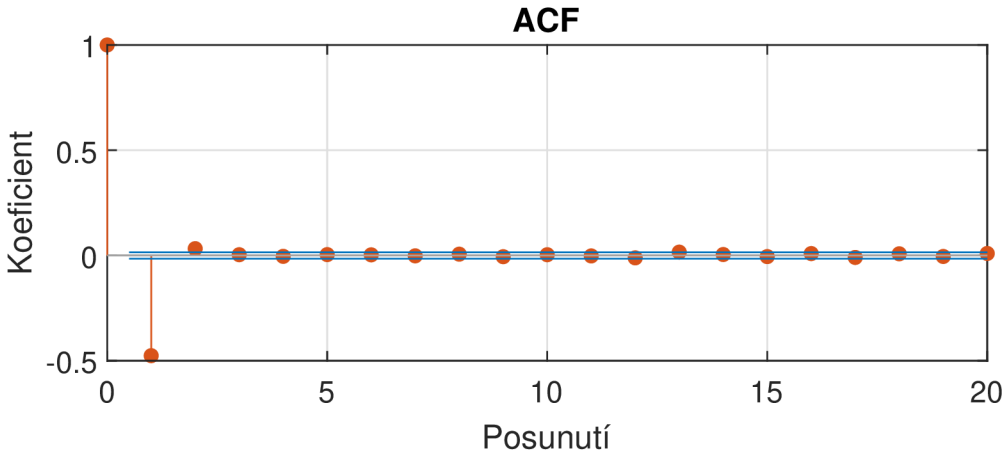
Odhad parametrů se provádí na základě ACF a PACF. Jedná se pouze o odhad, proto je potřeba vyzkoušet a vyhodnotit více variant modelu s různými parametry ve fázi **ověřování modelu**. [13]

2.2.7 Vyhodnocovací kritéria

Abychom dokázali posoudit rozdíl mezi skutečnou hodnotou a modelem je vhodné použít vyhodnocovací kritéria - např. MAE, MSE, RMSE, MAPE, AIC a BIC.

Střední absolutní chyba - MAE

$$MAE = \frac{1}{n} \sum_{t=1}^{n} |y_t - y_s| \tag{2.9}$$



Obr. 2.4: Ukázka korelogramu

Střední kvadratická chyba - MSE

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - y_s)^2 \tag{2.10}$$

Odmocnina střední kvadratické chyby - RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - y_s)^2} \tag{2.11}$$

Střední absolutní procentuální chyba - MAPE

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - y_s}{y_t}\right| \tag{2.12}$$

, kde n je počet prvků v časové řadě, kterou modelujeme, y_t je hodnota vypočítaná modelem, y_s je skutečná hodnota v daném kroku. MAE není vhodná, pokud máme příliš odlehlé hodnoty a máme porovnávat více časových řad. MSE je naopak vhodná pokud chceme penalizovat velké odchylky, její nevýhodou je, že ztrácí jednotky. RMSE řeší problém se ztrátou jednotek. MAPE je nezávislé na měřítku a je vhodné pro porovnání více modelů časových řad s různými měřítky. Penalizuje více negativní než pozitivní chyby, což vede k asymetrii. Nevýhodou MAPE je, že generuje nekonečné nebo nedefinované hodnoty pro nulové nebo téměř nulové hodnoty y_s. [17]

Akaikeho informační kritérium **AIC** a bayesovské informační kritérium **BIC** jsou kritéria používaná přímo k hodnocení kvality modelu a vypočítáme je následovně.

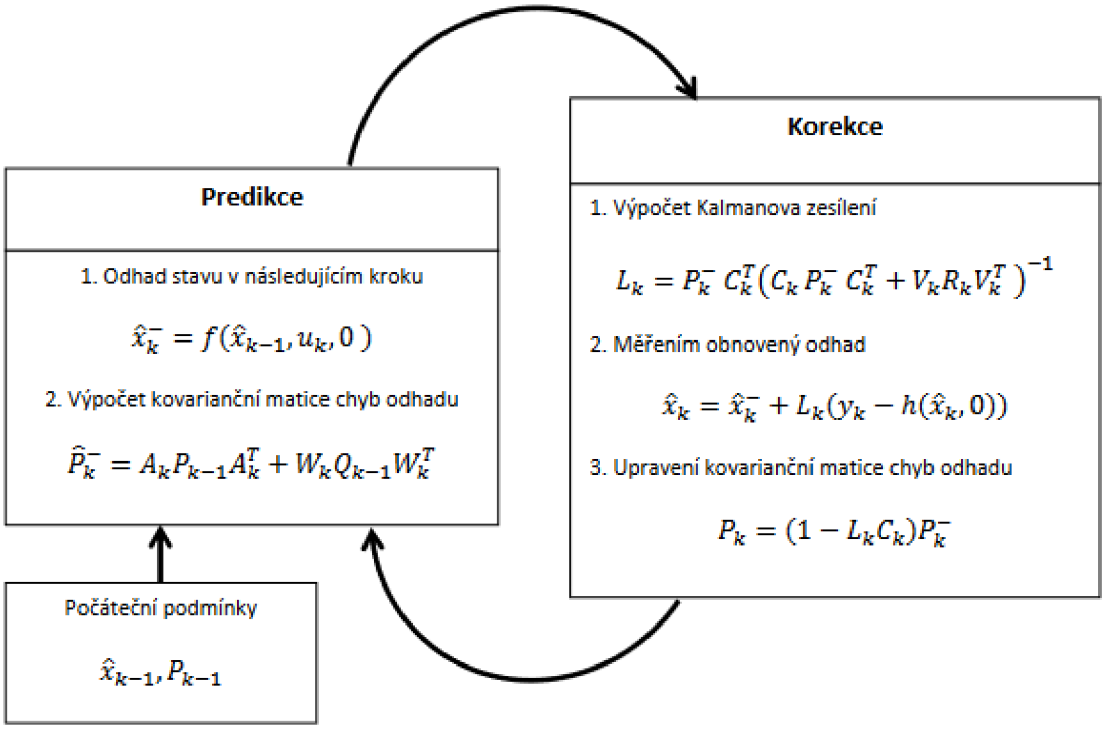
$$AIC = n \cdot ln(\sigma^2) + 2 \cdot (p + q) \tag{2.13}$$

$$BIC = ln(\sigma^2) + (p+q) \cdot ln(n) \tag{2.14}$$

, kde p, q jsou parametry modelu, σ je reziduální rozptyl použitého modelu a n je počet pozorování. [15]

2.2.8 Kalmanův filtr

Kalmanův filtr je algortimus, který slouží pro predikování a filtraci dat. Obrázek 2.5 znázorňuje průběh odhadu stavů. Odhad probíhá ve 2 krocích, predikci a korekci (zpřesnění) na základě měření. Kalmanův filtr lze použít na nelineární systémy, které linearizujeme rozvojem do Taylorovy řady - Rozšířený kalmanův filtr. [16]

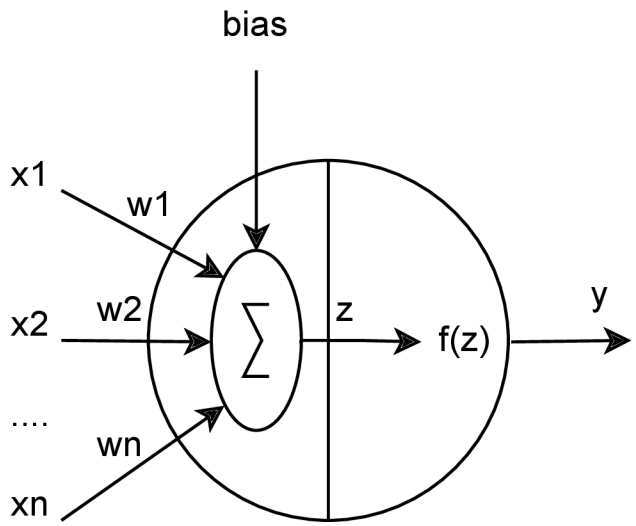


Obr. 2.5: Průběh odhadu Kalmanova filtru [18]

2.3 Umělé neuronové sítě

Umělé neuronové sítě (UNS) jsou matematickou analogií nervové soustavy (neuronové sítě) v mozku. Cílem neuronové sítě je na základě vstupního vektoru $X = (x_1, x_2, \ldots, x_n)$ získat požadovaný výstup Y. Základním prvkem UNS je matematický model neuronu, viz obrázek 2.6. Do každého neuronu vstupuje vektor X. Jednotlivé prvky vektoru jsou váhované váhami $W = (w_1, \ldots, w_n)$. Formální matematický popis neuronu vyjadřuje rovnice $y = f(z)$, kde

$$z = bias + \sum_{j=1}^{n} x_j \cdot w_j \tag{2.15}$$

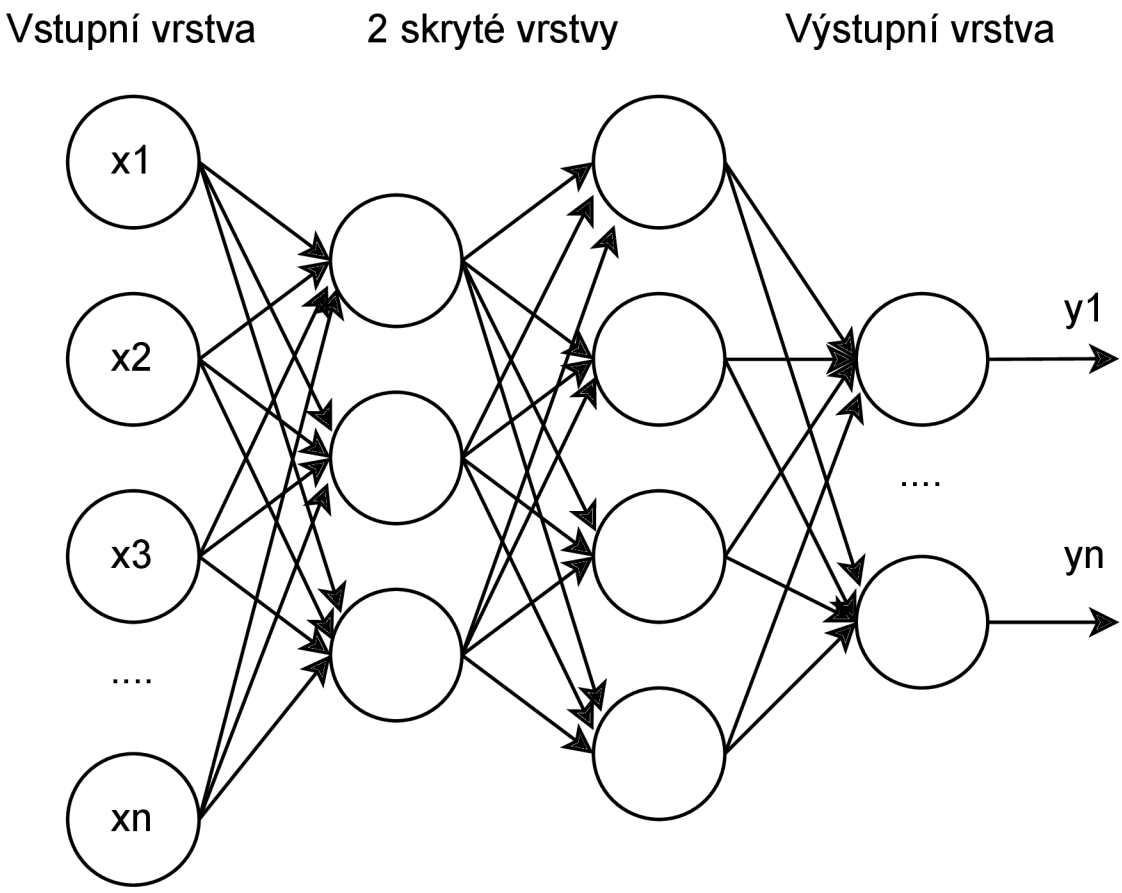


Obr. 2.6: Matematický model neuronu

Jednotlivé neurony jsou obvykle poskládány do neuronové sítě. Na obrázku 2.7 je ukázka jednoduché dopředné neuronové sítě, která se skládá ze 4 vrstev. První vrstvou je vstupní vrstva, která pouze distribuuje vstupní vektor X do druhé, tzv. skryté vrstvy, která je propojena s další skrytou vrstvou. Poslední vrstvou je vrstva výstupní.

Topologie, respektive architektura sítí nebo propojení neuronů v UNS jsou různé. Liší se vzájemným propojením neuronů, počtem neuronů v jednotlivých vrstvách, počtem vrstev (na obrázku 2.7 lze vidět 2 skryté vrstvy) a aktivačními funkcemi neuronů f(z).

Aby neuronové sítě správně generovaly požadovaný výstup, tak je potřeba je tzv. naučit.



Obr. 2.7: Ukázka dopředné neuronové sítě se 4 vrstvami (Vstupní (distribuční), 2 skryté a jedna výstupní)

Topologie (typy) neuronových sítí

Jak již bylo řečeno, neuronové sítě se skládají ze vstupní, skryté a výstupní vrstvy. Sítě lze rozdělit na **dopředné neuronové sítě** a **sítě se zpětnou vazbou** Vrstvy dopředných neuronových sítí jsou poskládány za sebe tak, jak lze vidět na obrázku 2.7. Výstup jedné vrstvy je připojen na vstup následující vrstvy. Neuronové sítě se zpětnou vazbou (rekurentní) mají výstupní vrstvu připojenu zpět na vstup.

Aktivační funkce neuronů

Existuje mnoho aktivačních funkcí. Nejznámější funkce jsou v tabulce 2.1. Nejoblíbenější aktivační funkcí byla sigmoida, která se také např. používá v logistické regresi k převodu lineární funkce na pravděpodobnost mezi 0 a 1. V dnešní době je preferovaná aktivační funkce ReLu (Rectified Linear unit), protože je efektivnější k výpočtu než sigmoida a není tolik náročná na paměť. [19]

Aktivační funkce	Matematický zápis
Sigmoida	$f(x) = \frac{1}{1+e^{-x}}$
Tangens hyberbolický	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Binární funkce	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$
ReLu	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$

Tab. 2.1: Nejpoužívanější aktivační funkce

Učení neuronových sítí

Učení se dělí na 2 způsoby, tzv. učení s učitelem nebo bez učitele. Algoritmus **Back propagation** (BP) je způsob učení s učitelem, kdy základem je trénovací množina, ve které jsou ke vstupním datům přiřazena požadovaná data výstupní. Cílem algoritmu je hledat a nastavit vstupní váhy W do jednotlivých vrstev tak, aby byla celková chyba sítě E_c minimální.

Proces učení pomocí algoritmu BP probíhá v následujících krocích:

1. Inicializace vah - málá náhodná čísla, např. v intervalu <-0,3; 0,3>
2. Předložení tréninkového vzoru
3. Výpočet výstupu sítě
4. Výpočet chyby sítě E_s (Chyba sítě v aktuálním kroku)
5. Test, jestli byly předloženy všechna tréninková data
6. Výpočet celkové chyby E_c
7. Test, jestli je E_c menší než požadovaná chyba
8. Úprava (adaptace) vah

2.4 Rešerše

Ve zdroji [20] se autoři zabývají souvislostmi mezi počasím a dopravní kongescí mezi rezidenční a turistickou oblastí vzdálené od sebe přibližně 200 km. Použili model vícenásobné lineární regrese, využívající data o počasí z obou oblastí a data o kongesci z turistické oblasti. Pro vyhodnocení použili hodnotící kritérium MAPE a dosáhli celkové přesnosti 84,8%.

V [21] autoři navrhují ARIMA(0,1,3) model pro krátkodobou predikci a uvádějí, že by měl být použit pro detekci nehod v dopravě.

Ve zdroji [22] autoři porovnávají konvoluční neuronové sítě pro klasifikaci stavu dopravy a rozdělují ji na předpoveď kongesce a detekci anomálií v dopravě. Používají analýzu zvuku z dopravy. Jedna nahrávka má cca 4 sekundy a autoři anotovali 14255 nahrávek, kde 70% dat použili pro trénování UNS, 20% pro testování a 10% pro validaci. Porovnali 4 typy architektur konvolučních sítí. Přesnost sítí byla 92,7%, 92,7%, 92,8% a 93,7% a rychlost analýzy byla 398, 830, 1022 a 274 sekund.

Ve zdroji [23] autoři navrhují konvoluční neuronovou síť pro binární klasifikaci kongesce. Používají data z GPS ve městě Záhřeb v Chorvatsku. Aby se vyhnuli sezónním vlivům, nepoužívají data z letních prázdnin, kdy jezdí mnoho lidí na dovolenou. Výsledky svého návrhu konvoluční neuronové sítě porovnávají s rekurentní neuronovou sítí. Autoři uvádějí, že přesnost rekurentní sítě se pohybovala mezi 82 - 89 % a jejich síť má přesnost přes 90%.

Kvalitu (stupně) dopravy specifikuje norma ČSN 73 6110. Ta rozděluje dopravu na stupně A-F a odvozuje je od průměrné cestovní rychlosti. Cestovní rychlost je 50 km/h na sběrných komunikacích.[24]

- **Stupeň A: Dopravní tok je volný.** Velmi nízká hustota dopravy, bez velkého časového zdržení a průměrná cestovní rychlost je 90 % rychlosti volné.
- **Stupeň B: Nerušený provoz.** Průměrná cestovní rychlost je 70 % rychlosti volné.
- **Stupeň C: Ustálený provoz.** Schopnost manévrovat v dopravním prostoru je omezována. Průměrná rychlost je snížena na 50 % cestovní rychlosti.
- **Stupeň D: Provoz je ještě stabilní.** Průměrná cestovní rychlost klesá ke 40 %.
- **Stupeň E: Kapacita je naplněna.** Průměrná rychlost klesá ke 30 % cestovní rychlosti. Rozsáhlá tvorba front před kritickými křižovatkami.
- **Stupeň F: Úsek je přetížen.** Rychlost klesá pod 25 % cestovní rychlosti. Dochází k zastavování a popojíždění vozidel .

V textu [25] se zaměřují na posouzení kongesce u mýtných stanic. Používají 5 stupňů kvality dopravy, rozdělených podle indexu kongesce (viz tabulka 2.2), který počítají podle vztahu 2.17, kde ρ se počítá podle vztahu 2.16. V značí rychlost a V_f je rychlost volného toku, kterou počítají jako průměr za určité období.

$$\rho = \frac{V}{V_\mathrm{f}} \tag{2.16}$$

$$TPI = \begin{cases} 0 \; if \; \rho > 1 \\ \frac{2^*(1-\rho)}{0.3} \; if \; 0.7 < \rho \leq 1 \\ 2 + \frac{2^*(0.7-\rho)}{0.2} \; if \; 0.5 < \rho \leq 0.7 \\ 4 + \frac{2^*(0.5-\rho)}{0.1} \; if \; 0.4 < \rho \leq 0.5 \\ 6 + \frac{2^*(0.4-\rho)}{0.1} \; if \; 0.3 < \rho \leq 0.4 \\ 8 + \frac{2^*(0.3-\rho)}{0.3} \; if \; \rho \leq 0.3 \end{cases} \tag{2.17}$$

Tab. 2.2: Stav doprav dle [25]

Stav	**Index kongesce**
Plynulá doprava	0 ÷ 2
Občasné kongesce	2 ÷ 4
Mírná kongesce	4 ÷ 6
Větší kongesce	6 ÷ 8
Kongesce	8 ÷ 10

3 Poskytnutá data

V této části chci nejprve popsat a přiblížit poskytnutá data a jednotlivé lokality.

3.1 Popis lokalit

Testovací data, dále jen data, jsou ze 6 náhodně vybraných pražských MUR detektorů. Následující seznam je výpis jednotlivých lokalit s krátkým popisem, kde se data sbírají.

1. MUR_BB-PO - ulice K Barandovu - Barrandovská, směr Pražský okruh
2. MUR_BE-BA - Bělohorská, směr Bílá Hora
3. MUR_BG-BM - K Barrandovu - Geologická, směr Barrandovský most
4. MUR_PV-TP - Průmyslová - V Chaloupkách, směr Teplice
5. HP-CE-I - Švehlova (u křižovatky s Hostivařskou)
6. HP-CE-O - Švehlova (za odbočkou na parkoviště Squash)

Abych získal lepší představu o lokalitách, tak jsem si pomocí webu *mapy.cz* a funkce *Panorama* jednotlivé lokality „prošel" a zjišťoval kolik má daná lokalitu pruhů, jak je přibližně dlouhý měřený úsek, který jsem funkcí *ruční měření* změřil, jaká je na daném úseku maximální povolená rychlost a jestli se v úseku vyskytují odbočky.

Došel jsem k těmto informacím. V lokalitě MUR_BB-PO se jedná o úsek 700 metrů s maximální povolenou rychlostí 70 km/h a úsek je bez odboček.

V lokalitě MUR_BE-BA jsem naměřil úsek 200 metrů s max. rychlostí 50 km/h se dvěma odbočkami (jeden příjezd a jeden odjezd z jednosměrky), tyto odbočky budou pravděpodobně málo významné, jelikož se jedná o dvě odbočky do, respektive z obydlené oblasti. Pravý pruh je pouze pro vozidla IZS, TAXI, BUS a kola. Z toho usuzuji, že provoz v jednom pruhu měl být nižší než ve druhém.

V lokalitě MUR_BG-BM jsem naměřil úsek dlouhý 200 m s max. rychlostí 50 km/h a bez odboček. V lokalitě MUR_PV-TP jsem naměřil 350 m dlouhý úsek s max. rychlostí 70 km/h a také bez odboček. V lokalitě HP-CE-I, HP-CE-O jsem nenašel typické úsekové měření a nedokázal jsem určit, jak dlouhý je úsek. Jedná se o lokality cca 430 metrů od sebe, mezi nimi jsou dvě relativně málo významné odbočky na parkoviště. Tyto informace jsem shrnul do tabulky 3.1.

3.1.1 Ukázka dat

Data jsou uložena ve formátu CSV (Comma-Separated Values, hodnoty oddělené čárkami. A vypadají následovně:

Tab. 3.1: Shrnutí dat z jednotlivých lokalit

Označení	**Počet pruhů**	**Délka úseku**	**Povolená rychlost**	**Odbočky**
MUR_BB-PO	2	700 m	70 km/h	Ne
MUR_BE-BA	2	200 m	50 km/h	Ano
MUR_BG-BM	2	200 m	50 km/h	Ne
MUR_PV-TP	2	350 m	70 km/h	Ne
HP-CE-I	2	-	50 km/h	Ano
HP-CE-O	2	-	50 km/h	Ano

"mur";"lane";"fivemin_time";"fivemin_time_local";"detection_count";
"velocity_avg";"car_count";"lgt_count";"hvt_count";"bus_count";"oth_count"
"MUR_BG-BM";1;"2023-01-31 23:00:00";"2023-02-01 00:00:00";2;25;2;0;0;0;0
"MUR_BG-BM";1;"2023-01-31 23:05:00";"2023-02-01 00:05:00";0;;;;;;
"MUR_BG-BM";1;"2023-01-31 23:10:00";"2023-02-01 00:10:00";4;51;3;0;0;0;0
"MUR_BG-BM";1;"2023-01-31 23:15:00";"2023-02-01 00:15:00";4;52;4;0;0;0;0
"MUR_BG-BM";1;"2023-01-31 23:20:00";"2023-02-01 00:20:00";4;56;3;0;0;0;0
"MUR_BG-BM";1;"2023-01-31 23:25:00";"2023-02-01 00:25:00";3;51;3;0;0;0;0
"MUR_BG-BM";1;"2023-01-31 23:30:00";"2023-02-01 00:30:00";0;;;;;;

Lze si hned všimnout, že druhý a sedmý záznam obsahuje prázdné data. Podobných záznamů se v datech vyskytuje mnoho a budu se jim věnovat později.

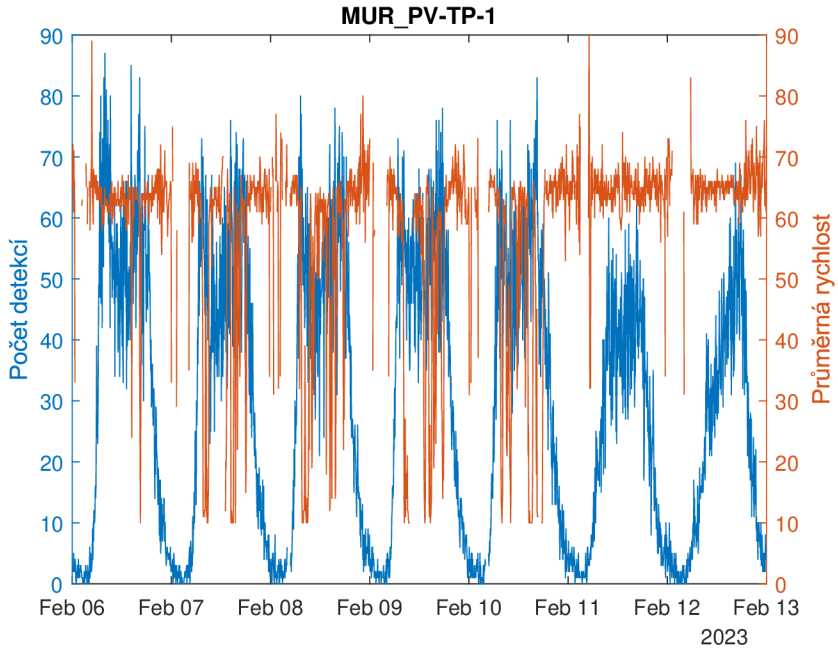
Data jsou za časové období 1.2.2023 - 31.3.2023 v 5 minutých intervalech. Poskytují informace o názvu detektoru (*mur*), čísla pruhu (*lane*), časové známce UTC a lokálním času (*fivemin_time* a *fivemin_time_local*), celkovém počtu zaznamenaných vozidel (*detection_count*), průměrné rychlosti (*velocity_avg*) a počtu vozidel v 5 kategoriích - osobní (*car_count*), lehká (*lgt_count*) a těžká vozidla (*hvt_count*), autobusy (*bus_count*) a ostatní vozidla (*oth_count*).

3.1.2 Grafická ukázka a detailnější popis dat

Na obrázku 3.1 jsou týdenní data z detektoru MUR_PV-TP-1.

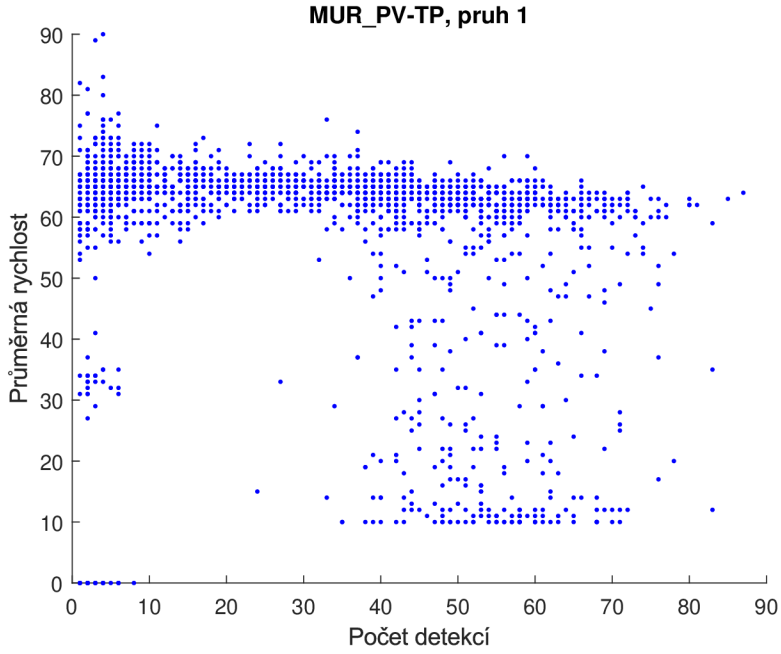
V pracovní dny kolem sedmé ráno lze vidět výrazný nárust v počtu detekcí, což odpovídá ranní špičce. Okolo poledne vidíme útlum v detekcích, který bych označil jako běžný provoz. Kolem páté odpoledne vidíme další výrazný nárust detekcí, který odpovídá odpolední špičce. O víkendech nenastávají ranní ani odpolední špičky a sledujeme běžný provoz. V nočních hodinách je počet detekcí minimální.

Podívejme se také na průměrnou rychlost v závislosti na počtu detekcí. V oblasti špiček lze vidět, že průměrná rychlost se snižuje s vyšším počtem detekcí, pokud je



Obr. 3.1: Ukázka týdenních dat z detektoru MUR_PV-TP-1, na levé ose vidíme počet detekcí, na pravé průměrnou rychlost vozidel

na určitém úseku mnoho vozidel, pak vozidla začínají zpomalovat. Na obrázku 3.2 je zřetelnější, že s vyšším počtem detekcí dochází ke snižování rychlosti, data jsou z detektoru MUR_PV-TP-1.



Obr. 3.2: Ukázka závislosti detekcí na rychlosti

4 Návrh a implementace řešení

V této části postupně navrhnu dílčí řešení s implementací v prostředí MATLAB R2023b a ukážu jakých výsledků jsem dosáhl. Rozhodl jsem se pro vyhledávání zajímavých situací v datech.

V prvním kroku bude třeba si data vhodně předzpracovat pro následnou jednodušší práci. Dále popíšu sadu nástrojů pro práci s daty, které budou uzpůsobeny pro konkrétní potřeby uživatele.

4.1 Příprava dat

Příprava dat zahrnuje načítání dat ze souboru a jejich třídění.

4.1.1 Načtení dat

Data načítám pomocí funkce *LoadData()*. Funkce nejprve „prohledá" složky v adresáři, kde se nachází soubor *LoadData.m* a nalezne všechny soubory s příponou **.csv*. Na základě počtu nalezených CSV souborů vytvoří výstupní buňkové pole. Definuje import možnosti, jako jsou počty, názvy a datové typy proměnných. Toto je nutný krok, protože MATLAB jinak použije předdefinované možnosti načítání a to mi způsobovalo problémy při načítání, protože ne všechny soubory mají stejný formát zápisu dat. V souborech HP-CE-*.csv je použita proměnná *slice*, kdežto v ostatních souborech je použita proměnná *mur*. Dále v souboru MUR_BE-BA_1.csv je pro zápis data použit formát *dd.mm.yyyy h:mm*, ale u ostatních je použito *yyyy-mm-dd hh:mm:ss*. A navíc, MATLAB načítal datum jako datový typ *string*. Pomocí funkce *readtable* jsou načteny všechny data ze souborů do buňkového pole, funkce končí a je vráceno výstupní buňkové pole, ve kterém jedna buňka odpovídá jedné tabulce dat.

Na výpisu 4.1 kód funkce.

Výpis 4.1: Funkce *LoadData()* pro načtení dat z CSV souboru do tabulky

```
function data = LoadData()
    files = dir('*\*.csv');
    data = cell([numel(files), 1]);
    opts = delimitedTextImportOptions('NumVariables',11);
    opts.DataLines = [2 Inf];
    opts.Delimiter = ';';
    opts.VariableNames = {'mur', 'lane', 'fivemin_time', '
   fivemin_time_local', 'detection_count', 'velocity_avg', '
```

```
    car_count', 'lgt_count', 'hvt_count', 'bus_count', '
    oth_count'};
8     opts.VariableTypes = {'string', 'double', 'datetime', '
    datetime', 'double', 'double', 'double', 'double', 'double
    ', 'double', 'double'};
9     opts.MissingRule = 'fill';
10    for file = 1:numel(files)
11        f = fullfile(files(file).folder, files(file).name);
12        mur = readtable(f, opts);
13        data{file} = sortrows(mur, "fivemin_time_local", "
    ascend");
14    end
15 end
```

4.1.2 Výběr dat podle dne v týdnu

Funkci jsem nazval GetDataDayOfWeek a jejím úkolem je vybrat všechny záznamy, které byly pořízeny daný den v týdnu např. v pondělky. Tato funkce má 3 vstupní parametry. Parametr *data* je vstupní tabulka obsahující data, které chci filtrovat. Parametr *day* je datového typu string a obsahuje den v týdnu. Parametr *var* je proměnná rozhodující o tom jestli budu data filtrovat podle sloupce *fivemin_time* nebo *fivemin_time_local*.

Nejprve je zkontrolováno, jestli je *day* datového typu string a *var* je *fivemin_time* nebo *fivemin_time_local*. Následně jsou pomocí funkce dateshift vybrána všechna data, která odpovídají danému dni v týdnu. A na konec jsou pomocí logického indexování vybrány všechny záznamy, které odpovídají vybraným datům.

Výpis 4.2: Funkce *GetDataDayOfWeek(data,day,var)*

```
1 function dataOut = GetDataDayOfWeek(data, day, var)
2     if isstring(day) & (var == 'fivemin_time_local' | var ==
    'fivemin_time')
3         days = dateshift(data{:,var}, 'dayofweek', day);
4         dataOut = data(data{:,var} == days, :);
5     end
6 end
```

4.1.3 Třídění dat podle dne v týdnu

Účelem funkce *SortCellArrayByDay* je roztřídit výstupní data z funkce *LoadData*. Vstupním parametrem je buňkové pole. Využívá funkci *GetDataDayOfWeek* pro vý-

sorted

12x7 cell

	1	2	3	4	5	6	7	8
1	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
2	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
3	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
4	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
5	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
6	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
7	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
8	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
9	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
10	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
11	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
12	384x10 table	384x10 table	432x10 table	432x10 table	432x10 table	384x10 table	384x10 table	
13								

Obr. 4.1: Ukázka výstupu z funkce SortCellArrayByDay

běr dat podle dne v týdnu a ukládá je do výstupní proměnné, kde řádky jsou jednotlivé data z MUR detektorů a sloupce odpovídají dnům v týdnu (1 ≈ pondělí).

Toto třídění jsem zvolil, abych mohl jednodušeji porovnávat data v jednotlivých dnech. Takto uspořádané data mohu snadno indexovat.

Výpis 4.3: Funkce *SortCellArrayByDay(data)*

```
function mursByDay = SortCellArrayByDay(data)
    if iscell(data)
        days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
    'Friday', 'Saturday', 'Sunday'];
        mursByDay = cell([numel(data), numel(days)]);
        for mur = 1:numel(data)
            for day = 1:7
                mursByDay{mur, day} = GetDataDayOfWeek(data{
   mur}, days(day), 'fivemin_time_local');
            end
        end
    end
end
```

4.1.4 Časová agregace

Další funkcí je *Resample*. Vstupy do funkce jsou buňkové pole, které je výstupem z načítací funkce nebo funkce *SortCellArrayByDay* a časový interval v minutách,

který odpovídá nové agregaci. Data jsou v 5 minutových intervalech a účelem této funkce je vytvořit jinou časovou agregaci, např. po 15 minutách.

Funkce nejprve zkontroluje, jestli je vstupní parametr buňkové pole. Poté předalokuje výstupní proměnnou. Pokud vstupní časový interval není datového typu duration, který je podmínkou pro převzorkování funkcí *retime*, kterou používám, je vstupní parametr přetypován. Dále jsou tabulky v buňkovém poli převedeny na časové tabulky. Pro převzorkování používám *retime*, ve které je argumentem, mimo jiné, metoda, jakou chci použít pro slučování dat. Pro sloučení jednotlivých počtu detekcí jsem zvolil sumu hodnot a pro sloučení průměrných rychlostí jsem zvolil průměr. Dále pak probíhá převedení zpět do tabulky a uložení do výstupního buňkového pole. Celá funkce je ve výpisu 4.4.

Výpis 4.4: Funkce *Resample*

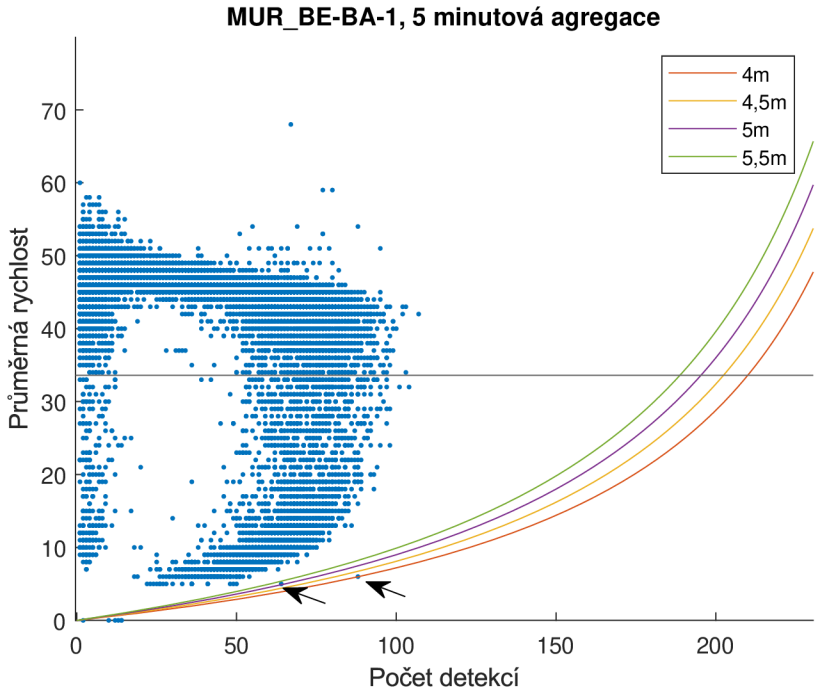
```
function outData = Resample(cellArray, minute)
    if iscell(cellArray)
        outData = cell(size(cellArray));
        if isduration(minute) == false
            minute = minutes(minute);
        end

        for i = 1:numel(cellArray)
            data = cellArray{i};
            TT1 = array2timetable(data{:, ["detection_count",
     "car_count", "lgt_count", "hvt_count", "bus_count" , "
   oth_count" ]} ...
                , 'RowTimes', data.fivemin_time_local, ...
                'VariableNames',["detection_count",  "
   car_count", "lgt_count", "hvt_count", "bus_count" , "
   oth_count" ]);
            TT2 = array2timetable(data{:, "velocity_avg"}, '
   RowTimes', data.fivemin_time_local, 'VariableNames',"
   velocity_avg");

            aggregatedCounts = retime(TT1,'regular', 'sum', '
   TimeStep',minute);
            aggregatedVelocity = retime(TT2,'regular', 'mean'
   , 'TimeStep',minute);
            aggregatedCounts.velocity_avg = round(
   aggregatedVelocity.velocity_avg);
            aggregatedCounts(:,["mur", "lane"]) = data(1:
   height(aggregatedCounts), ["mur", "lane"]);
```

```
19
20              outData{i} = timetable2table(aggregatedCounts);
21              outData{i} = renamevars(outData{i}, 'Time', '
    fivemin_time_local');
22          end
23      end
24  end
```

4.2 Kontrola fyzické konzistence

Kontrola konzistence vyplývá z textu [12], ve kterém autoři porovnávají maximální počet detekcí v závislosti na délce vozidla a rychlosti. Data, které jsou „pod křivkou" filtrují. Maximální počet detekcí získáme podle vztahu 4.1. Hodnotu je nutné vydělit příslušným číslem podle agregace dat (hodnotou 12 pro 5 min agregaci). Na obrázku 4.2 je vynesena vzájemná závislost rychlosti a počtu detekcí. Dále jsou vyneseny křivky maximálního počtu detekcí v závislosti na rychlosti a délky vozidla. Šipkou jsou označeny dva body, které nesplňují podmínku.

$$max\_pocet = \frac{prumerna\_rychlost[km/h] \cdot 1000}{delka\_vozidla + bezpecna\_vzdalenost} \tag{4.1}$$



Obr. 4.2: Závislost rychlosti na počtu detekcí s maximálním počtem detekcí

4.3 Stavy detektoru

Jak jsem v kapitole 3.1.1 zmínil, v datech se vyskytují chybné záznamy. Toho lze využít při návrhu detekce stavu detektoru. Analyzoval jsem jaké nastávají situace chybějících záznamů a co by to mohlo znamenat.

4.3.1 Výpadek detektoru

Jednou ze situací je, kdy senzor nezaznamenal žádnou hodnotu (NaN). Ta může být způsobena tím, že nedošlo k detekci, nebo že došlo k výpadku. V datech se však vyskytují záznamy, které jsou nulové. Můžu tedy tvrdit, že pokud se v několika sloupcích vyskytuje *NaN*, došlo k výpadku.

Z tohoto důvodu jsem napsal funkci *DetectNConsecutiveNaNs*, která slouží pro hledání N po sobě jdoucích *NaN* hodnot. Lze také zvolit proměnné, ve kterých se vyhledává.

Funkce postupně prochází pole a kontroluje jestli N předešlých prvků nemá hodnotu, pro každou zvolenou proměnnou. Pokud všech N prvků ve všech zvolených proměnných nemá hodnotu označí je log. 1. Výstupem z funkce je logické pole, ve kterém jsou log. 1 označeny všechny záznamy, které mají N po sobě jdoucích *NaN*. Logické pole pak můžeme použít pro klasifikaci stavu výpadku. Záměrně neuvádím, jestli se jedná o krátkodobý nebo dlouhodobý výpadek. Závisí to na volbě N a časové agregaci. Zvolím např. N = 5 při 5 minutové agregaci, to odpovídá časovému úseku 25 minut. Při stejném N, ale 15 minutové agregaci se jedná o 75 minut. Ze znalosti N, nelze rozhodnout o jaký výpadek se jedná a je nutné znát informaci při jaké časové agregaci se daný vzorek posuzuje.

Výpis 4.5: Funkce *DetectNConsecutiveNaNs*

```
1
2  function out = DetectNConsecutiveNaNs(data, vars, N)
3      out = zeros(height(data), 1, 'logical');
4      k = 0:N-1;
5      for i = N:height(data)
6          tmp = zeros(1, numel(vars));
7          for j = 1:numel(vars)
8              tmp(j) = all(isnan(data{i-k, vars(j)}));
9          end
10         if all(tmp)
11             out(i-k) = 1;
12         end
13     end
14 end
```

4.3.2 Výpadek měřicí stanice

Dalším typem výpadku je výpadek celé stanice, který se vyznačuje tím, že v datech chybí záznam. Tento výpadek detekuji pomocí funkce *CheckTimes*. Vstupy do funkce jsou tabulka dat, počáteční a koncové datum a velikost časové agregace. Na základě agregace a dat (množné číslo slova datum) je vygenerován časový vektor, který je následně porovnáván se vstupními daty. Chybějící data vrácena z funkce vč. indexu, kam patří. Pro správnou funkci je nezbytné přidat informaci o časovém pásmu, aby bylo zajištěno správné generování časového vektoru vzhledem k posunům mezi letním a zimním časem.

Výpis 4.6: Funkce *CheckTimes*

```
function [missing , i] = CheckTimesLocal(data, firstDate,
   lastDate, aggregation)
    firstDate.TimeZone = 'Europe/Prague';
    lastDate.TimeZone = 'Europe/Prague';
    data.fivemin_time_local.TimeZone = 'Europe/Prague';
    rightValues = firstDate:minutes(aggregation):lastDate;
    [missing, i] = setdiff(rightValues, data{:,'
   fivemin_time_local'});
    missing = missing';
    missing.TimeZone = '';
end
```

4.3.3 Výpadek detekce rychlosti

V datech se objevují záznamy, které mají nenulový počet vozidel a nemají uvedenou rychlost. Tyto záznamy hledám pomocí funkce *DetectNaNVelocity*. Ta funguje na stejném principu počítání N po sobě jdoucích NaN záznamů jako funkce *DetectNConsecutiveNaNs*. Rozdíl je v tom, že první zmíněná funkce zahrnuje i kontrolu správnosti detekce vozidel. Pokud je detekce nulová, pak rychlost může být NaN.

Výpis 4.7: Funkce *DetectNaNVelocity*

```
function out = DetectNaNVelocity(data, N)
    tmp = zeros(height(data), 1, 'logical');
    out = tmp;
    k = 0:N-1;
    tmp(isnan(data{:,'velocity_avg'}) & data{:,'
   detection_count'} > 0) = 1;
    for i = N:height(tmp)
        if sum(tmp(i-k)) == N
```

```
            out(i-k) = 1;
        end
    end
end
```

4.3.4 Zaseknutí detektoru

Dalším stavem detektoru je zaseknutí. Tento stav hledá funkce *IsDetectorStucked* - výpis 4.8. Ta postupně prochází N-tice záznamů a porovnává jejich hodnoty, pokud jsou všechny stejné, záznamy označí.

Výpis 4.8: Funkce *IsDetectorStucked*

```
function out = IsDetectorStucked(data, N, vars)
    out = zeros(height(data),1,'logical');
    k = 0: N-1;
    for i = N:height(data)
        for j = 1: numel(vars)
            same(j) = compareElements(data{i-k,vars(j)});
        end
        if all(same)
            out(i-k) = 1;
        end
    end
end

function result = compareElements(array)
    firstElement = array(1);
    result = all(array == firstElement);
end
```

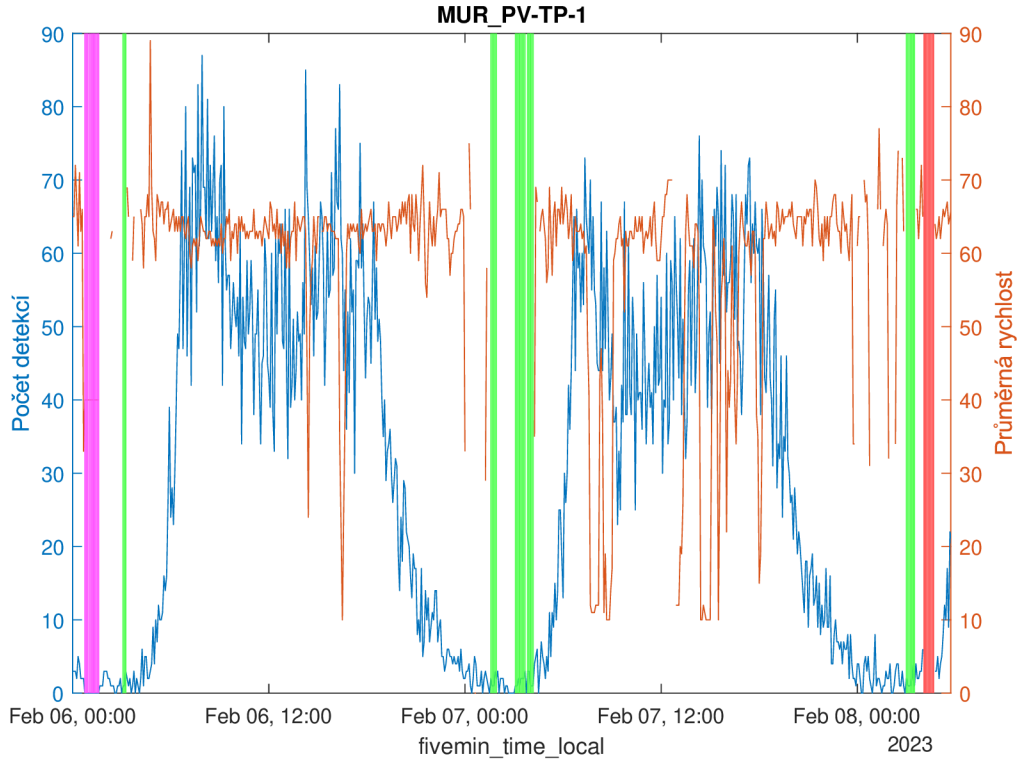
4.3.5 Ukázka detekce výpadků

Pro detekci výpadků jsem zvolil náhodně detektor MUR_PV-TP-1. Jelikož se nikde nevyskytoval špatný výpočet rychlosti, tyto data jsem nasimuloval. U 10 záznamů jsem změnil počet detekcí na 0 a průměrnou rychlost na 40.

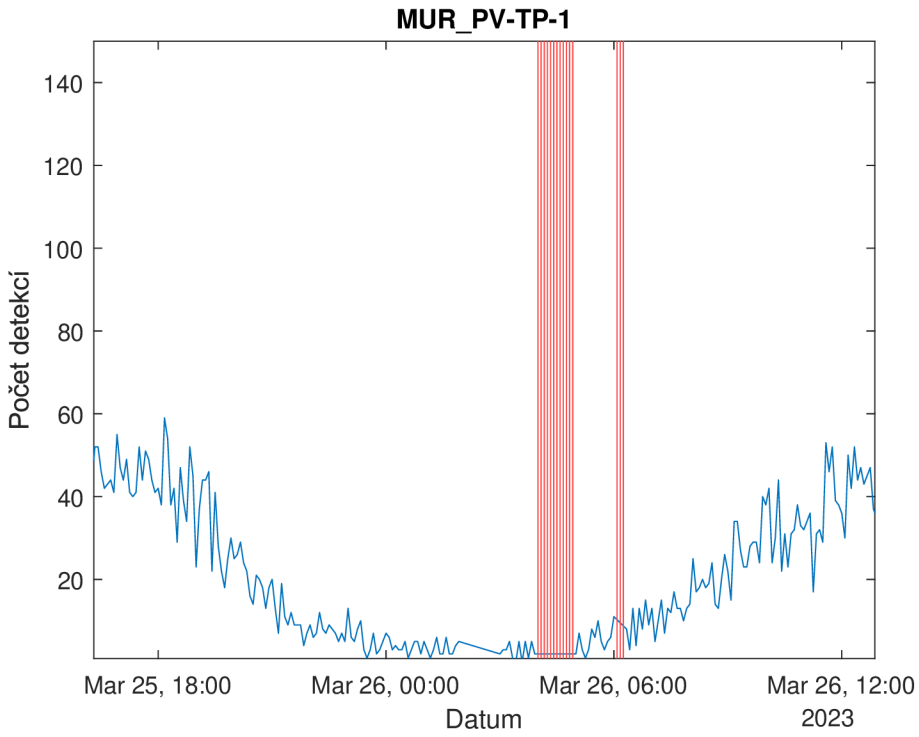
Na obrázku 4.3 jsou fialově označeny data se špatným výpočtem rychlosti, zeleně jsou výpadku měření rychlosti a červeně jsou výpadky detektoru. Zvolil jsem N = 5 při 5 min agregaci.

Na obrázku 4.4 je ukázka výpadku celé stanice MUR_PV-TP-1. Záznamy, které chybí jsou označeny červeně.

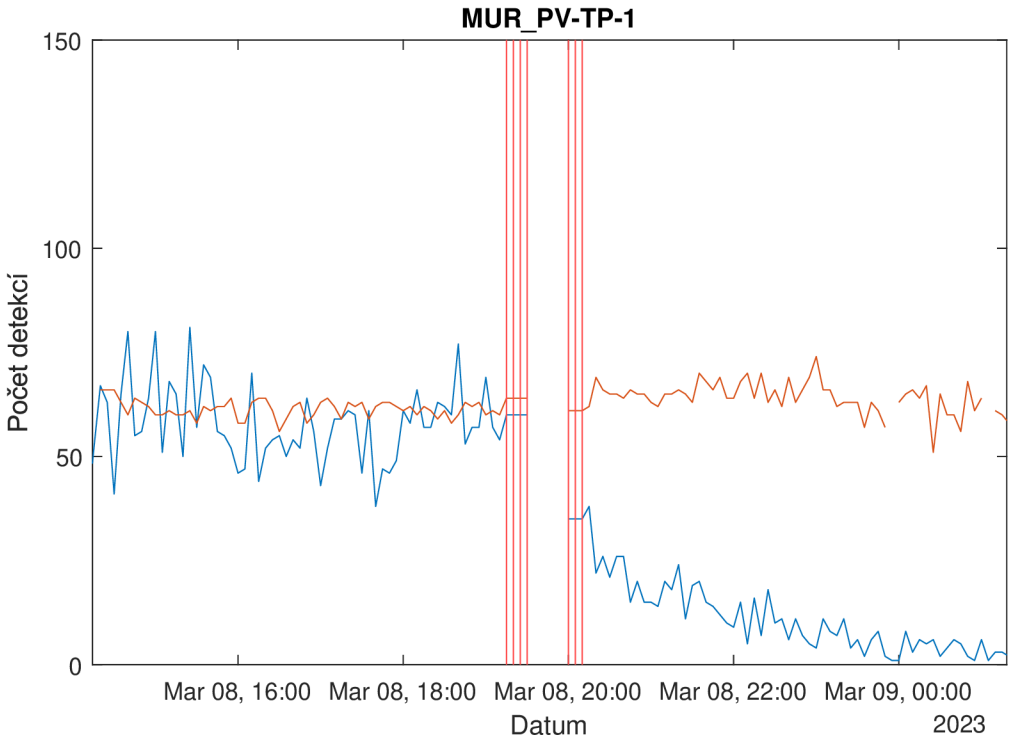
Na obrázku 4.5 lze vidět červeně označená data, na který jsem detekoval zaseknutý detektor. N jsem zvolil 3.



Obr. 4.3: Ukázka detekce výpadků 1, fialově špatný výpočet rychlosti, zeleně výpadek rychlosti a červeně výpadek detektoru, N = 5, 5 minutová agregace



Obr. 4.4: Ukázka detekce výpadku stanice - červeně, 5 minutová agregace



Obr. 4.5: Ukázka zaseknutí stanice MUR_PV-TP-1 - červeně, 5 minutová agregace

4.4 Stavy dopravy

4.4.1 Norma ČSN 73 6110

Jedním z návrhů určení stavu dopravy je podle normy ČSN 73 6110, viz kapitola Rešerše. V kapitole 2.4 jsem podrobněji popsal stupně dopravy A-F podle této normy. Norma uvádí, že rychlost volného toku sběrných komunikací je 50 km/h. Definuji následující rozsahy rychlostí pro určení stupně dopravy. Například pro stupeň B je to 70 % z rychlosti volného toku, tedy 70 % z 50 km/h je 35 km/h. Pro stupeň C je to 25 km/h. Prostřední hodnota mezi 25 km/h a 35 km/h bude hranice pro přechod mezi stupni B a C.

- **Stupeň A: Volný tok.** Rychlost 40 km/h a více.
- **Stupeň B: Nerušený provoz.** 30 ÷ 40 km/h.
- **Stupeň C: Ustálený provoz.** 23 ÷ 30 km/h.
- **Stupeň D: Provoz je ještě stabilní.** 16 ÷ 23 km/h.
- **Stupeň E: Kapacita je naplněna.** 13 ÷ 16 km/h
- **Stupeň F: Úsek je přetížen.** 13 km/h a méně

V MATLABu pak implementace vypadá následovně:

Výpis 4.9: Funkce *norma*

```
function mur = norma(mur) % A-F ~ 1-6
out = zeros(height(mur), 1);
out(mur.velocity_avg >= 40) = 1;
out(mur.velocity_avg < 40 & mur.velocity_avg >= 30) = 2;
out(mur.velocity_avg < 30 & mur.velocity_avg >= 23) = 3;
out(mur.velocity_avg < 23 & mur.velocity_avg >= 16) = 4;
out(mur.velocity_avg < 16 & mur.velocity_avg >= 13) = 5;
out(mur.velocity_avg < 13) = 6;
mur.out = out;
end
```

4.4.2 Index kongesce

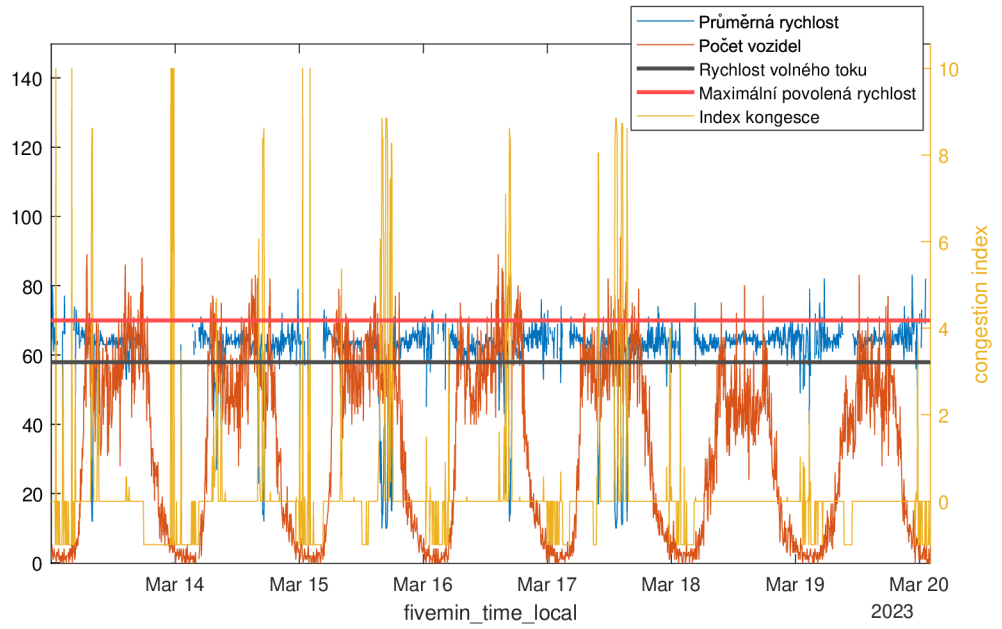
Na základě textu [25] implementuju výpočet indexu kongesce (vztah 2.17) pro poskytnutá data. Nejprve je nutné spočítat rychlost volného toku. Podle textu je to průměr rychlostí všech vozidel od 6:00 do 22:00. Poté pro každý záznam je vypočítáno ρ a index. Pokud záznam nemá hodnotu nebo je nulová, index má hodnotu -1.

Na obrázku 4.6 jsou vynesené hodnoty detekcí, průměrných rychlostí a indexu kongesce pro detektor MUR_PV-TP-1. Lze vidět, že podle indexu dochází ke kon-

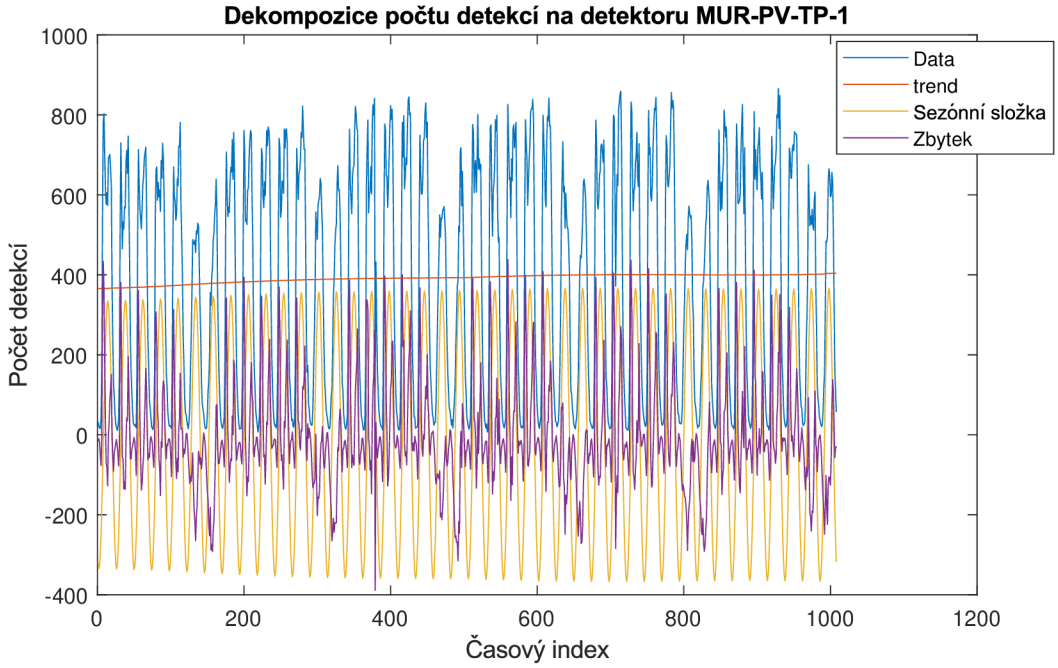
gescím spíše v odpoledních špičkách. TO lze sledovat i na rychlosti, že dochází k propadu. Během noci dochází více k výpadkům.

Výpis 4.10: Funkce *StateIdentificationBasedOnSpeed*

```
function mur = StateIdentificationBasedOnSpeed(mur)
    newMur = GetDataHourRange(mur, 6, 22, 'fivemin_time_local');
    Vf = mean(newMur.velocity_avg,'omitmissing');
    mur.congestion_index = SpecificCongestionIndex(mur.velocity_avg ./ Vf)
end

function TPI = SpecificCongestionIndex(ro)
    TPI = zeros(size(ro));
    mask1 = ro > 1;
    mask2 = 0.7 < ro & ro <= 1;
    mask3 = 0.5 < ro & ro <= 0.7;
    mask4 = 0.4 < ro & ro <= 0.5;
    mask5 = 0.3 < ro & ro <= 0.4;
    mask6 = ro <= 0.3;

    TPI(mask1) = 0;
    TPI(mask2) = 2 * (1 - ro(mask2)) / 0.3;
    TPI(mask3) = 2 + (2 * (0.7 - ro(mask3)) / 0.2);
    TPI(mask4) = 4 + (2 * (0.5 - ro(mask4)) / 0.1);
    TPI(mask5) = 6 + (2 * (0.4 - ro(mask5)) / 0.1);
    TPI(mask6) = 8 + (2 * (0.3 - ro(mask6)) / 0.3);
    TPI(~(mask1 | mask2 | mask3 | mask4 | mask5 | mask6 | ro == 0)) = -1;
end
```



Obr. 4.6: Výstup z funkce *StateIdentificationBasedOnSpeed*



Obr. 4.7: Dekompozice časové řady detekce vozidel stanice MUR_PV-TP-1 s 60 minutovou agregací

5 Modelování časové řady

Nyní už mám připravené data a nástroje pro určení stavů dopravy a detektorů. Rozdělím si data na trénovací a testovací. Šest týdnů použiji jako trénovací a jeden týden jako testovací. Dalším krokem je namodelovat si řadu a předpovědět, jakým způsobem se bude řada vyvíjet. Pro modelování používám vždy stejná data a 60 minutovou agregaci, aby byly výsledky porovnatelné.

5.1 Dekompozice časové řady

Analyzovat/modelovat časovou řadu můžeme pomocí dekompoziční metody. Řadu můžeme rozdělit na reziduum, trendovou a sezónní složku pomocí příkazu *trenddecomp()*. Pomocí reziduí poté spočítám hodnotící kritéria, ty pro časovou agregaci 60 minut nabývají hodnot: MAE = 584.5179, MSE = 109220, RMSE = 330.4785, MAPE = 195.4166. Pro časovou agregaci 5 minut pak jsou hodnoty kritérií následující: MAE = 58.1226, MSE = 1081.2, RMSE = 32.8820, MAPE = 231.3274.

5.2 SARIMA model

Časová řada je periodická, tzn. je nestacionární a je potřeba ji diferencovat. Sezóna modelu záleží na časové agregaci časové řady. Vytvářím týdenní model, takže sezónu vypočítám jako $7 \cdot 24 \cdot \frac{60}{\text{časová agregace v minutách}}$. Podle ACF se odhaduje parametr p a podle PACF parametr q.

Dívám se na hodnoty funkcí ACF a PACF a čísla *Lagů*, u kterých začnou být hodnoty funkcí subjektivně nevýznamné jsou odhadnuté parametry. V příloze A na obrázku A.1 jsou vykreslené ACF a PACF grafy diferencované řady detekce vozidel. Parametr p odhaduji na 1 a parametr q odhaduji mezi 1 a 4. Jelikož se jedná o odhadnuté parametry, musím vhodnost modelu ověřit pomocí hodnotících kritérií.

Vytvořím modely s různými parametry p, q, P, Q. Pro každý model spočítám koeficienty a předpovím týdenní data. Na základě předpovězených dat vypočítám hodnoty MAE, MSE, RMSE, MAPE. Jakmile spočítám koeficienty všech modelů, jsou pro každý model spočítány kritéria AIC a BIC.

Kód programu je v příloze E, výpis E.1.

5.2.1 Týdenní SARIMA modely

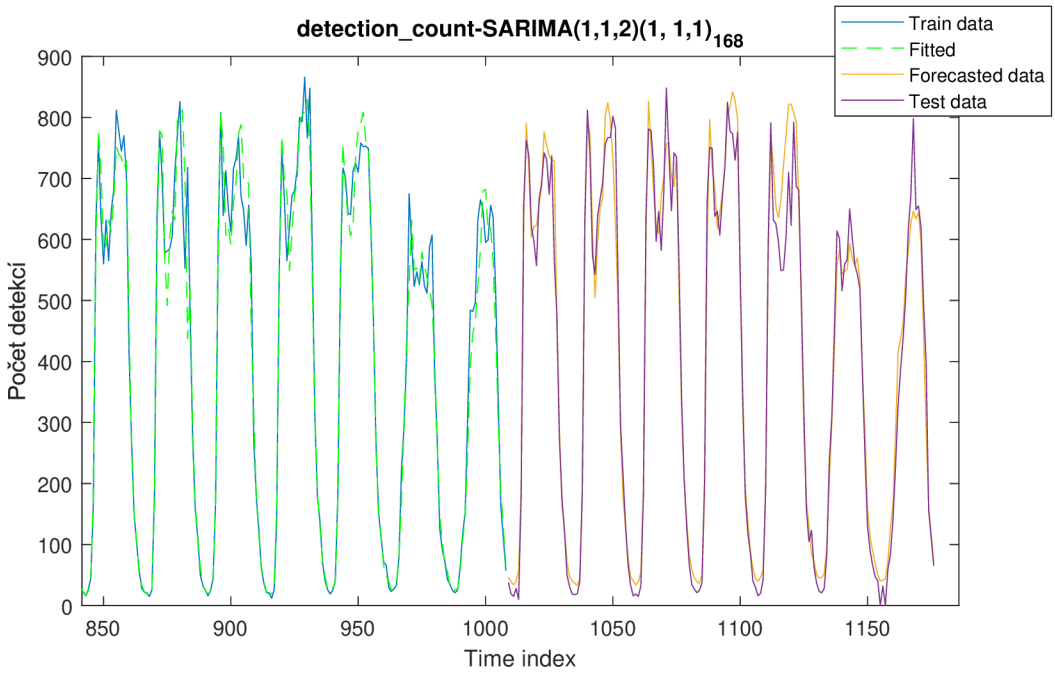
Celkem jsem vytvořil 16 modelů s různými parametry. Hodnoty kritérií a parametrů p, q, P, Q pro modely počtu detekcí jsou v tabulce B.1 v příloze B. Nejlépe vychází

model
$SARIMA(1,1,2)(1,1,1)_{168}$, který má 5 ze 6 nejmenších kritérií (zvýrazněno tučně).

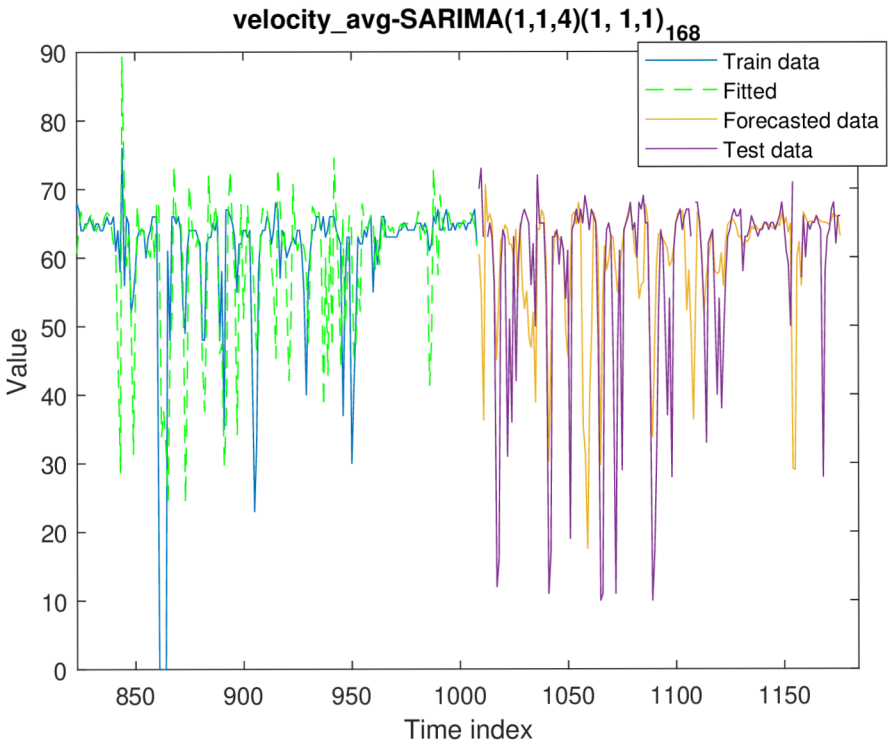
Výstupy z tohoto modelu jsou na obrázku 5.1. Jedná se o týdenní model a byl vytvořen z dat ze stanice MUR_PV-TP-1 s agregací 60 minut.

Stejným postupem jsem vytvořil i modely pro průměrné rychlosti.

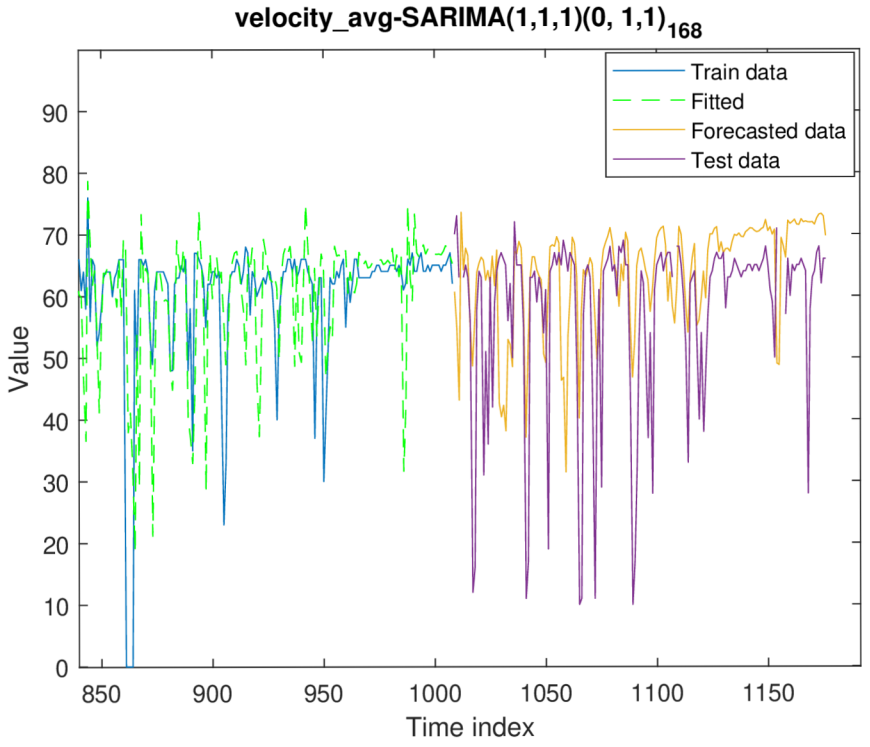
Hodnoty kritérií a parametrů *p, q, P, Q* pro modely průměrných rychlostí jsou v tabulce B.2 v příloze B. Pro fitování modelu podle kritérií AIC a BIC vychází nejlépe
$SARIMA(1,1,4)(1,1,1)_{168}$. Predikce a fitování modelu je na obrázku 5.2. Pro účely predikce podle hodnot MAE, MSE, RMSE je nejvhodnější $SARIMA(1,1,4)(0,1,1)_{168}$ (obrázek 5.3).



Obr. 5.1: $SARIMA(1,1,2)(1,1,1)_{168}$ model detekce počtu vozidel



Obr. 5.2: $SARIMA(1,1,4)(1,1,1)_{168}$ model průměrné rychlosti

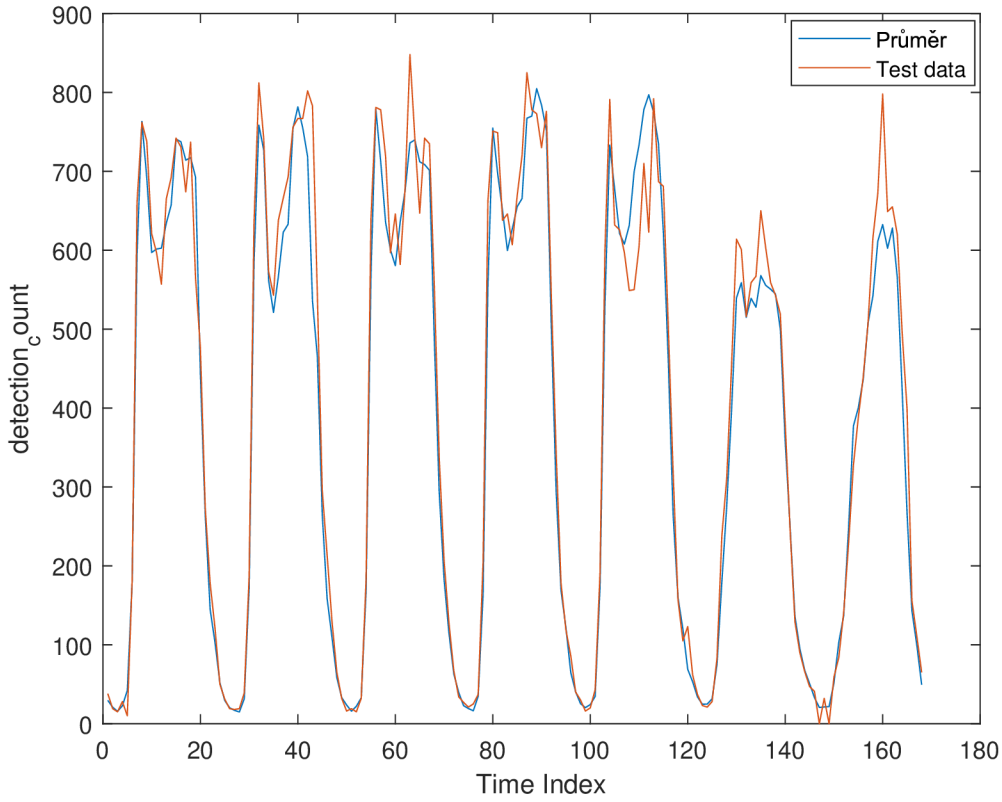


Obr. 5.3: $SARIMA(1,1,4)(0,1,1)_{168}$ model průměrné rychlosti

5.3 Model na základě průměrů časů mezi týdny

Model vytvořím tak, že pro každý den (např. každé pondělí) ve stejný čas spočítám průměr a standardní odchylku. Kód je v příloze C, výpis C.1. Porovnám model s testovacími daty (průběhy vykresleny na obrázku 5.4) a spočítám hodnoty kritérií: MAE = 32.6905, MSE = 2483.8, RMSE = 49.8377, MAPE = 12.8648.

V příloze D na obrázku D.1 jsou průběhy jednotlivých dnů v týdnu, fialově je průměr a čerchovaně je standardní odchylka pro daný čas. V příloze D, obrázek D.2 je detail průběhů pro pondělí.



Obr. 5.4: Porovnání modelu průměru s testovacími daty

Závěr

V první kapitole v úvodní části definuji dopravní veličiny. Dále se pak věnuji indukčním smyčkám, kamerovým systémům, radarům a FCD, u kterých vysvětlím, na jakém principu fungují a jak sbírají data. Druhá kapitola je o metodách analýzy dopravních dat. Fyzické konzistenci a metodám na základě historických dat. Ve třetí kapitole popisuji formát a grafické zobrazení dopravních dat. „Co lze z grafu vyčíst." Ve čtvrté kapitole navrhuji a implementuji funkce v MATLABU pro načítání dat, jejich třídění filtraci a pro samotnou klasifikaci stavů dopravy a detektorů.

Pro ověření správnosti klasifikace stavů jsem data ručně prošel. Buď jsem si vykreslil graf nebo jsem procházel jednotlivé záznamy a sledoval, jestli funkce správně detekují všechny stavy, které jsem navrhl. Toto řešení ověřování není ideální a je velmi zdlouhavé. Nicméně, jsem si poměrně jistý, že dané stavy detekuji správně.

V další kapitole jsem modeluji časové řady pomocí Box-Jenkinsovy metody. Porovnávám vhodnost modelů mezi sebou na trénovacích a testovacích datech. Z takto porovnaných dat vyplynulo, že model $SARIMA(1,1,2)(1,1,1)_{168}$ je nejvhodnější pro modelování dopravního toku na daných trénovacích datech. Časová náročnost výpočtu je velmi vysoká a s každým dalším parametrem, který bych chtěl otestovat, exponenciálně narůstá.

Vytvořil jsem také model na základě průměrů mezitýdenních hodnot ve stejný čas. Porovnám model s testovacími daty a vypočítám kritéria: MAE = 32,6905, MSE = 2483,8, RMSE = 49,8377, MAPE = 12,8648. Porovnám kritéria se $SARIMA(1,1,2)(1,1,1)_{168}$ modelem, který má hodnoty kritérií: MAE= 34,079, MSE = 2353,061, RMSE = 48,508, MAPE = 16,022. Lze si všimnout, že hodnoty těchto modelů pro predikci na základě těchto kritérií jsou porovnatelné a nelze rozhodnout, který model je lepší. SARIMA model je výpočetně náročnější, jednodušší na implementaci, avšak model s průměry je mírně složitější na implementaci, ale samotný výpočet je pak mnohem rychlejší.

Literatura

[1] HRUBÝ, Filip. *Posouzení možností využití různých predikčních modelů pro dostupná dopravní data.* Diplomová práce. Praha: České učení technické v Praze, 2023.

[2] SINGH, Niraj Kumar; VANAJAKASHI, Lelitha a TANGIRALA, Arun K. *Segmentation of vehicle signatures from inductive loop detector (ILD) data for real-time traffic monitoring.* Online. IEEE Xplore. 2018. Dostupné z: `https://ieeexplore.ieee.org/document/8328281`. [cit. 2024-01-04].

[3] JAGADEESH, George R.;DHINESH, George R. a SRIKANTHAN Thambipillai *Method for accuracy assessment of aggregated freeway traffic data.* Online. Institution of Engineering and Technology. 2014. Dostupné z: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-its.2013.0094. [cit. 2023-12-22].

[4] HALLIDAY, David; RESNICK, Robert; WALKER, Jearl; DUB, Petr; KOMRSKA, Jiří et al. Fyzika. Přeloženo 5. vydání. Vysoké učení technické v Brně - Nakladatelství VUTIUM a PROMETHEUS Praha, 2000. ISBN 80-214-1869-9.

[5] TEXAS INSTRUMENTS. *Fundamentals of FMCW Radar.* Online. Texas Instruments developer zone. C2023. Dostupné z: `https://dev.ti.com/tirex/explore/node?node=A__AXNV8Pc8F7j2TwsB7QnTDw__RADAR-ACADEMY__GwxShWe__LATEST&placeholder=true`. [cit. 2024-01-04].

[6] CACERES, Noelia; M. ROMERO, Luis; G. BENITEZ, Francisco a M. DEL CASTILLO, Jose. *Traffic Flow Estimation Models Using Cellular Phone Data.* Online. IEEE Xplore. 2012. Dostupné z: https://ieeexplore.ieee.org/document/6172682. [cit. 2023-12-19].

[7] GISGEOGRAPHY. *How GPS Receivers Work – Trilateration vs Triangulation.* Online. GIS Geography. 2022. Dostupné z: https://gisgeography.com/trilateration-triangulation-gps/. [cit. 2023-12-21].

[8] CHANG, Ande; JIANG, Guiyan a NIU, Shifeng. *Traffic Congestion Identification Method Based on GPS Equipped Floating Car.* Online. IEEE Xplore. 2010. Dostupné z: https://ieeexplore.ieee.org/document/5523297. [cit. 2023-12-21].

[9] *Fundamental diagrams*. Online. -. Dostupné z: https://ocw.tudelft.nl/wp-content/uploads/Chapter-4.-Fundamental-diagrams.pdf. [cit. 2024-01-04].

[10] HANČLOVÁ, Jana a TVRDÝ, Lubor. *Úvod do analýzy časových řad*. Ostrava: Vysoká Škola Báňská -Technická Univerzita Ostrava, 2003. Dostupné také z: https://www.fd.cvut.cz/department/k611/PEDAGOG/VSM/7_AnalyzaCasRad.pdf.

[11] KŘIVÝ, Ivan. *Analýza časových řad*. Ostrava: Ostravská univerzita v Ostravě, 2012. Dostupné také z: https://publi.cz/download/publication/20?online=1.

[12] BACHECHI, Chiara, Federica ROLLO a Laura PO. Detection and classification of sensor anomalies for simulating urban traffic scenarios. Cluster Computing [online]. 2022, 25(4), 2793-2817 . ISSN 1386-7857. Dostupné z: doi:10.1007/s10586-021-03445-7. [cit. 2024-05-21]

[13] KÖPPELOVÁ, Jana. Modely časových řad podnikových ukazatelů. Dizertační práce. Praha: ČZU v Praze, 2019.

[14] HAYES, Adam, STAPLETON, Chip a MUNICHIELLO, Katrina (ed.). *Autoregressive Integrated Moving Average (ARIMA) Prediction Model*. Online. Investopedia. 2023. Dostupné z: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp. [cit. 2024-01-04].

[15] PETRÁŠKOVÁ, Vladimíra. *Prognostické modely v oblasti modelování finančních časových řad*. Dizertační práce. Praha: Česká zemědělská univerzita v Praze, 2006.

[16] FRONČKOVÁ, Kateřina. *Kalmanovy filtry*. Diplomová práce. Hradec Králové: Univerzita Hradec Králové, 2018. Dostupné také z: https://theses.cz/id/vxef9f/STAG89197.pdf.

[17] RINK, Konstantin. *Time Series Forecast Error Metrics You Should Know*. Online. Medium. 2021. Dostupné z: https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27. [cit. 2024-01-04].

[18] VESELÝ, Libor. *Algortimy bezsnímačového řízení synchronního motoru s permanentními magnety*. Dizertační práce. Brno: VUT v Brně, 2013. Dostupné také z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=62615.

[19] GARETH, James; HASTIE, Trevor; WITTEN, Daniela a TIBSHIRANI, Robert. *An Introduction to Statistical Learning*. 2. 2021. Dostupné také z: https://www.karlin.mff.cuni.cz/~pesta/NMFM334/StatLearning/Book2nd/ISLRv2_website.pdf.

[20] HONG, Bonghee; LEE, Kyungmin; LEE, Jiwan a JANG, Yang-Ja. *A Prediction Model of Traffic Congestion Using Weather Data*. Online. IEEE Xplore. 2016. DostupnÃľz:https://ieeexplore.ieee.org/document/7396485. [cit. 2024-01-04].

[21] AHMED, Mohamed S. a COOK, Honolulu Allen R. *Analysis of Freeway Traffic Time-Series Data by Using Box-Jenkins Techniques*. Oklahoma: University of Oklahoma, 1979.

[22] BUI, Khac-Hoai Nam; YI, Hongsuk a OH, Hyeonjeong. *Traffic Density Classification Using Sound Datasets: An Empirical Study on Traffic Flow at Asymmetric Roads*. Online. IEEE Xplore. 2020. Dostupné z: https://ieeexplore.ieee.org/document/9136653. [cit. 2024-01-04].

[23] TIŠLJARIĆ, L; CARIĆ, T; ERDELIĆ, T a ERDELIĆ, M. *Traffic State Estimation Using Speed Profiles and Convolutional Neural Networks*. Online. IEEE Xplore. 2020. Dostupné z: https://ieeexplore.ieee.org/document/9245177. [cit. 2024-01-04].

[24] ČSN 73 6110. *Projektování místních komunikací*. Český normalizační institut, 2006.

[25] J. Liu, Q. Guo and G. Sheng, *Traffic State Identification for Expressway Network Based on ETC Gantry System*. Online. IEEE Xplore. Dostupné z doi: 10.1109/ICCCBDA51879.2021.9442498. [cit. 2024-05-21]

Seznam symbolů a zkratek

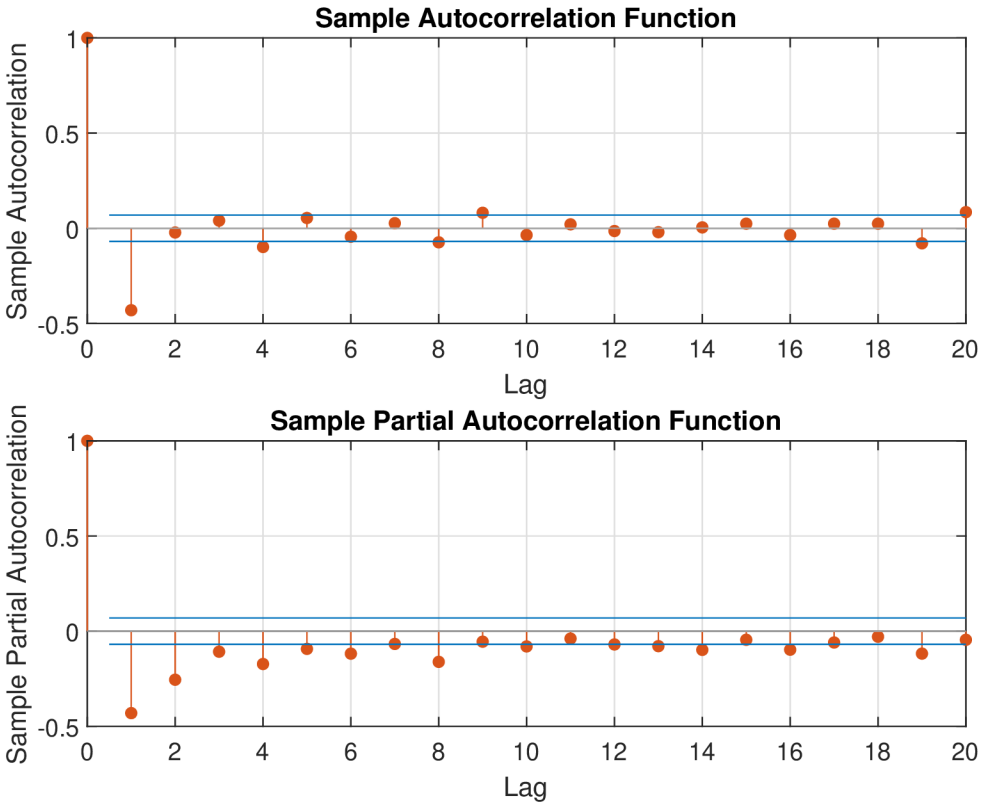
ITS	Intelligent transport system - inteligentní dopravní systém
MUR	Měření úsekové rychlosti
KTDS	Komplexní telematický a dohledový systém
FCD	Floating Car Data - data z plovoucích vozidel
GNSS	Globální navigační systém
AR	Autoregresní model
MA	Moving average - Proces klouzavých průměrů
ARIMA	Autoregresive integrated moving average - autoregresní integrovaný klouzavý průměr
SARIMA	Seasonal autoregresive integrated moving average - sezónní autoregresní integrovaný klouzavý průměr
ACF	Autocorrelation function - autokorelační funcke
PACF	Parcial autocorrelation function - parciální autokorelační funcke
MAE	Mean absolut error - střední absolutní chyba
MSE	Mean squared error - střední kvadratická chyba
RMSE	Root mean squared error - odmocnina střední kvadratická chyba
MAPE	Mean absolute percentage error - střední absolutní procentuální chyba
AIC	Akaike information criterion - Akaikeho informační kritérium
BIC	Bayesian information criterion - Bayesovské informační kritérium
UNS	Umělé neuronové sítě
ReLu	Rectified Linear unit
BP	Back propagation
IZS	Integrovaný záchranný systém
CSV	Comma separated values - hodnoty oddělené čárkami

UTC Coordinated Universal Time - koordinovaný světový čas

NaN Not a Number - „nečíslo”, žádná hodnota

Seznam příloh

A ACF a PACF diferencované řady



Obr. A.1: ACF a PACF diferencované časové řady

B Tabulka hodnot kritérií SARIMA modelů

Tab. B.1: Porovnání SARIMA týdenních modelů počtu detekcí s různými parametry

Model	AIC	BIC	MAE	MSE	RMSE	MAPE
SARIMA(1,1,1)(0,1,0)	10,740	10,759	47,718	4083,549	63,903	20,272
SARIMA(1,1,1)(0,1,1)	10,499	10,523	36,241	2492,363	49,924	18,166
SARIMA(1,1,1)(1,1,0)	10,511	10,535	35,752	2586,323	50,856	16,594
SARIMA(1,1,1)(1,1,1)	10,474	10,503	35,481	2470,257	49,702	17,370
SARIMA(1,1,2)(0,1,0)	10,729	10,753	44,709	3791,595	61,576	18,511
SARIMA(1,1,2)(0,1,1)	10,489	10,518	34,753	2358,903	48,569	16,999
SARIMA(1,1,2)(1,1,0)	10,497	10,527	34,195	2467,707	49,676	**14,500**
SARIMA(1,1,2)(1,1,1)	**10,463**	**10,497**	**34,079**	**2353,061**	**48,508**	16,022
SARIMA(1,1,3)(0,1,0)	10,729	10,758	44,828	3803,623	61,674	18,574
SARIMA(1,1,3)(0,1,1)	10,490	10,524	34,773	2361,998	48,600	17,004
SARIMA(1,1,3)(1,1,0)	10,498	10,532	34,287	2475,051	49,750	14,648
SARIMA(1,1,3)(1,1,1)	10,464	10,503	34,095	2355,786	48,536	16,024
SARIMA(1,1,4)(0,1,0)	10,724	10,758	46,725	3981,509	63,099	19,740
SARIMA(1,1,4)(0,1,1)	10,491	10,530	34,780	2366,523	48,647	16,977
SARIMA(1,1,4)(1,1,0)	10,499	10,538	34,416	2483,016	49,830	14,851
SARIMA(1,1,4)(1,1,1)	10,466	10,510	35,244	2453,792	49,536	17,088

Tab. B.2: Porovnání SARIMA týdenních modelů průměrných rychlostí s různými parametry

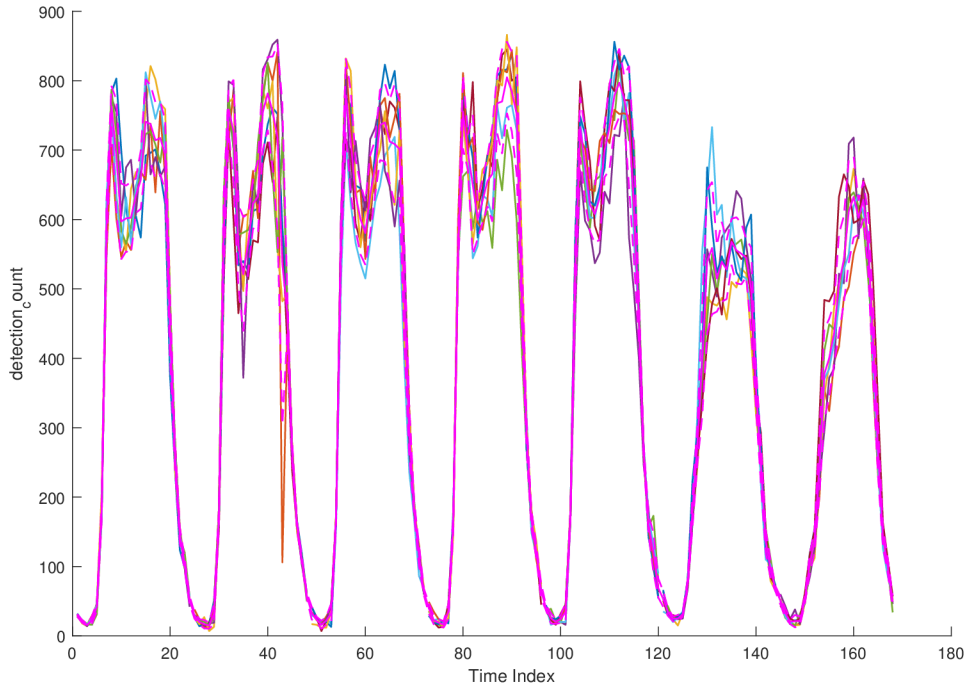
Model	**AIC**	**BIC**	**MAE**	**MSE**	**RMSE**	**MAPE**
SARIMA(1,1,1)(0,1,0)	8,079	8,099	10,699	329,739	18,159	68,089
SARIMA(1,1,1)(0,1,1)	7,788	7,813	9,876	213,700	14,618	17,362
SARIMA(1,1,1)(1,1,0)	7,747	7,772	9,290	240,581	15,511	20,507
SARIMA(1,1,1)(1,1,1)	7,717	7,746	9,913	219,439	14,813	18,819
SARIMA(1,1,2)(0,1,0)	8,081	8,105	10,568	327,782	18,105	70,053
SARIMA(1,1,2)(0,1,1)	7,787	7,816	9,371	204,301	14,293	16,750
SARIMA(1,1,2)(1,1,0)	7,749	7,779	9,297	240,689	15,514	20,514
SARIMA(1,1,2)(1,1,1)	7,717	7,752	9,678	215,132	14,667	18,512
SARIMA(1,1,3)(0,1,0)	8,069	8,098	9,321	308,728	17,571	200,977
SARIMA(1,1,3)(0,1,1)	7,765	7,799	7,724	179,432	13,395	**15,057**
SARIMA(1,1,3)(1,1,0)	7,734	7,768	8,789	234,240	15,305	20,328
SARIMA(1,1,3)(1,1,1)	7,696	7,735	8,164	193,229	13,901	16,992
SARIMA(1,1,4)(0,1,0)	8,067	8,102	9,302	307,341	17,531	266,748
SARIMA(1,1,4)(0,1,1)	7,763	7,802	**7,669**	**177,888**	**13,337**	15,096
SARIMA(1,1,4)(1,1,0)	7,730	7,769	8,403	228,195	15,106	20,662
SARIMA(1,1,4)(1,1,1)	**7,691**	**7,735**	8,021	190,335	13,796	17,088

C Kód pro výpočet průměrů dnů v týdnu

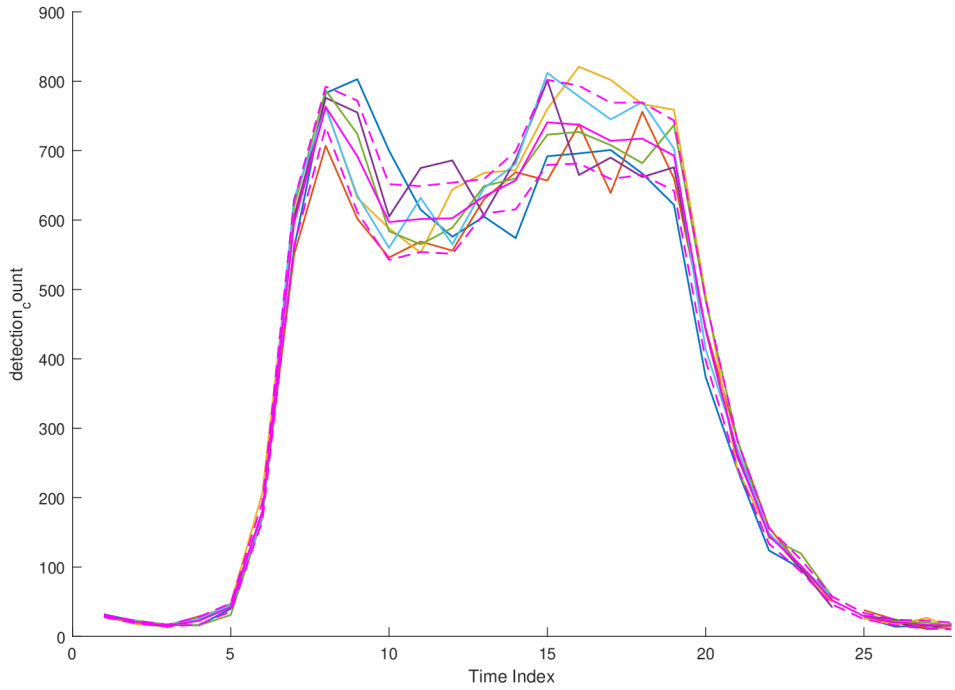
Výpis C.1: Kód pro vytvoření modelu průměru

```
murs = SortCellArrayByDay(trainData)
val = 24*60/aggregation;
for d = 1:7
    indx = 1+(d-1)*val : val +(d-1)*val;
    data = murs{mur, d}; %d = day, Monday = 1
    uniqueDays = unique(dateshift(data.fivemin_time_local, 'start', 'day'));

    dailyCountData = NaN(val,numel(uniqueDays));
    dailySpeedData = dailyCountData;

    for i = 1:length(uniqueDays)
        currentDay = day(uniqueDays(i));
        currentMonth = month(uniqueDays(i));
        currentYear = year(uniqueDays(i));

        dD = data{day(data.fivemin_time_local) == currentDay & month(data.fivemin_time_local) == currentMonth  year(data.fivemin_time_local) == currentYear, ['detection_count' 'velocity_avg ']};

        dailyCountData(1:height(dD), i) = dD(:,1);
        dailySpeedData(1:height(dD), i) = dD(:,2);

    end
    allCountData{d} = dailyCountData;
    allSpeedData{d} = dailySpeedData;

    average = [average;mean(dailyCountData,2, 'omitmissing')];
    error = [error;std(dailyCountData,0, 2, 'omitmissing')];
end
```

D Detail pondělí pro model průměru



Obr. D.1: Průběhy jednotlivých dnů mezi týdny a fialově průměr s čerchovanou standardní odchylkou



Obr. D.2: Detail pondělků pro model průměru

E Program pro tvorbu a testování SARIMA modelů

Výpis E.1: Kód pro tvorbu a testování vhodnosti a predikce SARIMA modelů

```
allData = LoadData()
aggregation = 60;
seasonality = 7*24*60/aggregation;
var = 'detection_count';
allData = Resample(allData, aggregation);

allTrainData = GetDataDayInterval(allData{7}, datetime('2023-02-06 00:00:00'), datetime('2023-03-19 23:55:00'));
allTestData = GetDataDayInterval(allData{7}, datetime('2023-03-20 00:00:00'), datetime('2023-03-26 23:55:00'));

trainData = allTrainData{:,var};
testData = allTestData{:,var};

d = 1;
i = 1;
p = 1;

for q = 1:4
    for P = 0:1
        for Q = 0:1
            Mdl(i) = arima('ARLags',1:p, 'D', d,'MALags', 1:q, 'Seasonality',seasonality , 'SARLags',seasonality*(1:P), 'SMALags',seasonality*(1:Q));
            Mdl(i).SeriesName = var;
            i = i+1;
        end
    end
end

numMdl = numel(Mdl);
for i = 1:numMdl
    [estMdl,~,logL(i)] = estimate(Mdl(i), trainData);
    results = summarize(estMdl);
    numParam(i) = results.NumEstimatedParameters;
    numObs(i) = results.SampleSize;
```

```
33      numForecasts = length(testData);
34      [forcastedValues, forecastedErrors] = forecast(estMdl,
    numForecasts, 'Y0', trainData);
35      estModels(i) = estMdl;
36      MAE(i) = sum(abs(forcastedValues - testData))/
    numForecasts;
37      MSE(i) = sum((forcastedValues - testData).^2)/
    numForecasts;
38      RMSE(i) = sqrt(MSE(i));
39      MAPE(i) = 100 * sum(abs((forcastedValues - testData) ./
    forcastedValues))/numForecasts;
40
41  end
42  [aic, bic] = aicbic(logL,numParam,numObs,Normalize=true)
```

F Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `logo` ..... loga školy a fakulty
  - `obrazky` ..... adresář obshující obrázky
  - `pdf` ..... pdf stránky generované informačním systémem
  - `text` ..... zdrojové textové soubory
  - `matlab` ..... adresář obsahující zdrojové soubory k matlabu a poskytnuté data
    - `data_mur_1` ..... adresář obsahující poskytnuté data
    - `data_mur_2` ..... adresář obsahující poskytnuté data
  - `sablona-prace.tex` ..... hlavní soubor pro sazbu kvalifikační práce
  - `thesis.sty` ..... balíček pro sazbu kvalifikačních prací