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

PREDIKCE DOJEZDOVÝCH DOB
TRAVEL TIME PREDICTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Andrej Mudroch

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Peter Honec, Ph.D.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Andrej Mudroch ***ID:*** 145557
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Predikce dojezdových dob

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh metody predikce dojezdových dob vozidel založený na umělé inteligenci, vytvoření testovacího modulu (performance & accuracy testy, ověření vhodností příznaků) a implementace modulu a vhodného rozhraní, které umožňuje zapojení do existující senzorické a výpočetní infrastruktury. K dispozici budou reálná data silničních senzorů z definovaných úseků.

1. Seznamte se s problematikou výpočtu a predikce dojezdových dob a používaných prediktorů založených na umělé inteligenci.
2. Navrhněte vhodné modely prediktorů.
3. Otestujte modely a proveďte rozbor platforem pro testy.

a. Výběr platformy.
b. Implementace testovaných modulů.
c. Rozbor a příprava vhodných příznaků.
d. Zhodnoťte vlastnosti modelů a jejich možná rizika.

4. Vytvořte interface prediktoru pro existující aplikaci a otestujte v reálných podmínkách.
5. Zhodnoťte práci a porovnejte se současným modelem.

DOPORUČENÁ LITERATURA:

Hlaváč, Šonka, Počítačové vidění.

Šonka, Hlávač, Boyle - IMAGE PROCESSING, ANALYSIS, AND MACHINE VISION,

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 15.5.2017

Vedoucí práce: Ing. Peter Honec, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Táto práca sa zaoberá predikciou dojazdových dôb vozidiel na cestných komunikáciách založenej na metódach strojového učenia. Je v nej rozobraná teória týkajúca sa dojazdových dôb a jednotlivé vedecké práce, ktoré sa venujú tejto problematike. V práci je vypracovaná analýza reálnych dát týkajúcich sa dojazdových dôb a sú navrhnuté príznaky, ktoré sú následne použité na tvorbu predikčných modelov. V rámci diplomovej práce bol navrhnutý a implementovaný komplexný predikčný systém, ktorého funkčnosť bola overená v praxi.

KĽÚČOVÉ SLOVÁ

dojazdové doby, predikcia, regresia, clustering

ABSTRACT

This thesis discusses travel time prediction of vehicles on roads based on the methods of machine learning. It describes theory of travel times and summarizes scientific papers dealing with this topic. Within the thesis, analysis of real travel time data was done and the features to be used in prediction models were engineered. Finally, the complex prediction system was designed and implemented and has been tested in production environment.

KEYWORDS

travel times, prediction, regression, clustering

MUDROCH, Andrej *Predikce dojezdových dob*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace, 2017. 92 s. Vedúci práce bol Ing. Peter Honec, PhD.

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovoú prácu na tému „Predikce dojezdových dob“ vypracoval(a) samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi Ing. Petrovi Honcovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

OBSAH

Úvod **12**

1 Dojazdové doby **13**
1.1 Zber dát pre dojazdové doby . . . 13
1.2 Definície dojazdových dôb . . . 14
1.2.1 Pravdivá priemerná dojazdová doba . . . 14
1.2.2 Priemerná dojazdová doba založená na príjazde . . . 14
1.2.3 Priemerná dojazdová doba založená na odjazde . . . 15
1.3 Rozdiely definícií dojazdových dôb v praxi . . . 16
1.4 Psychologické hľadisko predikcie dojazdových dôb a ich zobrazovania 17

2 Strojového učenie pre regresné problémy **18**
2.1 Regresné algoritmy . . . 18
2.1.1 Lineárna regresia . . . 18
2.1.2 Ridge Regression . . . 19
2.1.3 Lasso . . . 19
2.1.4 ElasticNet . . . 19
2.1.5 Support Vector Regression . . . 20
2.1.6 k-Nearest Neighbour . . . 21
2.1.7 Neurónové siete . . . 22
2.2 Ensemble learning . . . 23
2.2.1 Bagging . . . 24
2.2.2 Boosting . . . 24
2.3 Algoritmy data miningu . . . 25
2.3.1 K-Means . . . 26
2.3.2 DBSCAN . . . 28
2.4 Vyhodnocovacie kritériá . . . 29
2.4.1 MAE . . . 29
2.4.2 MAPE . . . 30
2.4.3 RMSE . . . 30
2.4.4 RMSLE . . . 30
2.4.5 R^2 . . . 30

3 Metódy predikcie dojazdových dôb **31**
3.1 Qiao, Haghani, Hamedi - 2013 [22] . . . 31

3.2 Lunde, Wolff - 2015 [5] . . . 32
3.3 Yildirimoglu, Geroliminis - 2013 [24] . . . 33
3.4 Elhenawy, Chen, Rakha - 2014 [6] . . . 35

4 Zdroje dát **37**
4.1 Senzorické dáta . . . 37
4.1.1 det1 . . . 37
4.1.2 det2 . . . 38
4.1.3 Okamžité dojazdové doby . . . 38
4.1.4 Spätne dopočítané dojazdové doby . . . 39
4.2 Chybné dáta . . . 39

5 Analýza reálnych dát **40**
5.1 Analýza dát zo senzorov . . . 40
5.1.1 det1 . . . 40
5.1.2 det2 . . . 41
5.2 Analýza okamžitých a reálnych hodnôt dojazdových dôb . . . 41

6 Návrh príznakov a predikčných modelov **48**
6.1 Výber vhodných príznakov a ich tvorba . . . 48
6.1.1 Tvorba príznakov . . . 48
6.1.2 Overovanie prediktívnej hodnoty príznakov . . . 50
6.1.3 Výber príznakov . . . 58
6.2 Predikčné modely . . . 58
6.2.1 TimeDomainModel . . . 58
6.2.2 ClusterModel . . . 59
6.2.3 CombinedModel . . . 59

7 Vývoj testovacieho frameworku **61**
7.1 Výber platformy . . . 61
7.2 Architektúra . . . 62
7.2.1 TestFramework . . . 62
7.2.2 plots . . . 63
7.2.3 benchmarks . . . 64

8 Porovnanie modelov **65**
8.1 Nájdenie parametrov modelov . . . 65
8.2 Testy modelov na úsekoch . . . 65
8.2.1 Úsek 1 . . . 66
8.2.2 Úsek 2 . . . 66

8.2.3 Úsek 3 . . . 67
8.3 Zhodnotenie modelov . . . 70

9 Predikčný systém **71**
9.1 Modul data_procesing . . . 72
9.2 Modul interface . . . 73
9.2.1 TrainingManager . . . 74
9.2.2 SectionInterface . . . 76
9.2.3 PredictionManager . . . 78
9.3 Modul models . . . 80
9.4 Modul support_files . . . 81
9.5 Modul configs . . . 81
9.6 Test predikčného systému . . . 82

10 Nastavenie predikčného systému **84**
10.1 Spustenie systému . . . 84
10.1.1 Spustenie s využitím platformy Docker . . . 84
10.1.2 Manuálne spustenie . . . 84

11 Záver **86**

Literatúra **87**

Zoznam symbolov, veličín a skratiek **91**

ZOZNAM OBRÁZKOV

1.1 Znázornenie pravdivej priemernej dojazdovej doby [1] 15
1.2 Znázornenie dojazdovej doby založenej na príjazde [1] 15
1.3 Znázornenie dojazdovej doby založenej na odjazde [1] 16
2.1 Znázornenie funkčnosti SVR 21
2.2 Perceptron [12] 23
2.3 Dopredná neurónová sieť [16] 24
2.4 Bagging - zlepšenie predikcie [17] 25
2.5 Iterácie algoritmu K-Means 27
2.6 Clusterizácia s využitím DBSCAN 29
3.1 Porovnanie predikcie dojazdových dôb - lasso, boosting a bagging z [5] 33
3.2 Príklad mapy zahltenia z [24] 35
3.3 Štruktúra predikčného modelu uvedeného v [22] 36
5.1 Rozloženie premennej **det1_velocity** pre jednotlivé senzorové rezy v úseku 42
5.2 Rozloženie premennej **det1_occupancy** pre jednotlivé senzorové rezy v úseku 43
5.3 Rozloženie premennej **det1_count** pre jednotlivé senzorové rezy v úseku 44
5.4 Rozloženie premennej **det2_count** pre jednotlivé senzorové rezy v úseku 45
5.5 Porovnanie rozloženia hodnôt dojazdových dôb pre víkendy a pracovné dni 46
5.6 Porovnanie priebehov dojazdových dôb pre jednotlivé dni v týždni . . 47
6.1 Porovananie okamžitej dojazdovej doby (**bck_tt**) a reálnej dojazdovej doby (**tt_real**) 50
6.2 Korelačná matica príznakov vyplývajúcich zo senzorov **det1** a výslednej dojazdovej doby 52
6.3 Korelačná matica príznakov vyplývajúcich zo senzorov **det2** a výslednej dojazdovej doby 53
6.4 Korelačná matica príznakov vyplývajúcich časových údajov a výslednej dojazdovej doby 54
7.1 Architektúra modulu **test_suite** 62
8.1 Porovnanie predikcií modelov počas rannej špičky na úseku 1 67
8.2 Porovnanie predikcií modelov počas rannej špičky na úseku 2 68
8.3 Porovnanie predikcií modelov počas rannej špičky na úseku 3 69
9.1 Štruktúra predikčného systému 71
9.2 Architektúra modulu **data_processing** 72

9.3 Architektúra tried na získavanie príznakov - **data_entities** 73
9.4 Architektúra modulu **interface** 74
9.5 Popis tvorby modelov implementovanej v triede **TrainingManager** . 75
9.6 Funkcionalita predikcie implementovaná triedou **SectionInterface** . 77
9.7 Popis princípu predikcie implementovanej v triede **PredictionManager** 79
9.8 Architektúra modulu **models** 80
9.9 Štruktúra modulu **support_files** 81

ZOZNAM TABULIEK

6.1 Výber príznakov korelovaných s **tt_real** 56
6.2 Signifikantné príznaky vybrané funkciou `feature_selection` 57
8.1 Porovnanie modelov aplikovaných na úsek 1 vzhľadom k reálnej dojazdovej dobe 66
8.2 Porovnanie modelov aplikovaných na úsek 1 vzhľadom k času potrebného na tréning a predikciu 66
8.3 Porovnanie modelov aplikovaných na úsek 2 vzhľadom k reálnej dojazdovej dobe 67
8.4 Porovnanie modelov aplikovaných na úsek 2 vzhľadom k času potrebného na tréning a predikciu 68
8.5 Porovnanie modelov aplikovaných na úsek 3 vzhľadom k reálnej dojazdovej dobe 68
8.6 Porovnanie modelov aplikovaných na úsek 3 vzhľadom k času potrebného na tréning a predikciu 69
9.1 Správy generované pri predikcii 78
9.2 Presnosť predikčného systému pri produkčnom teste - úsek 1 82
9.3 Presnosť predikčného systému pri produkčnom teste - úsek 2 82
9.4 Presnosť predikčného systému pri produkčnom teste - úsek 3 83

ÚVOD

Táto práca sa zaoberá návrhom metódy pre predikciu dojazdových dôb na definovaných úsekoch cestných komunikácií. Presné predikovanie dojazdovej doby dáva vodičom vozidiel možnosť si presnejšie naplánovať cestu a taktiež má významný psychologický efekt (pozitívny aj negatívny - v prípade nesprávne predikovanej, kratšej, dojazdovej doby ako je reálna). Sú tiež rozobrané a otestované možnosti a spoľahlivosť metód predikcie založených na regresných algoritmoch strojového učenia, ktoré v praxi preukázali svoju použiteľnosť v tejto problematike. Zadávateľom práce je firma CAMEA.

Prvá kapitola tejto práce poskytuje úvod do problematiky výpočtu a merania dojazdových dôb, definuje názvoslovie a potrebné termíny.

V druhej kapitole je uvedené zhrnutie metód a algoritmov používaných v strojovom učení. Sú popísané algoritmy SVR, lineárna regresia, Ridge a iné, ktoré preukázali svoju použiteľnosť pri praktickej predikcii dojazdových dôb. V rámci tejto kapitoly sú tiež uvedené štandardné kritériá kvality predikcie pri regresných úlohách.

Tretia kapitola je prehľadom výskumov a predchádzajúcich použití algoritmov strojového učenia na predikciu dojazdových dôb. Sú popísané modely s použitými príznakmi, ktoré boli k dispozícii v danom prípade, aby bolo možné nájsť zhodu s dodanými senzorickými údajmi, ak to bude možné.

Štvrtá kapitola sa venuje popisu zdrojov dát dodaných firmou CAMEA. Stručne sú popísané senzory, z ktorých dáta sú dostupné spolu s popisom údajov, ktoré poskytujú.

Piata kapitola sa venuje štatistickej analýze senzorických dát z jednotlivých úsekov. Sú znázornené rozloženia dát pre úseky a porovnané dojazdové doby pre jednotlivé úseky.

Šiesta kapitola sa venuje tvorbe príznakov zo senzorických dát. Je tiež overená prediktívna schopnosť jednotlivých príznakov. V rámci tejto kapitoly sú navrhnuté modely prediktorov, ktoré by bolo možné použiť pre predikciu dojazdových dôb.

Následne je v siedmej kapitole popísaná tvorba modulu pre testovanie a analýzu datasetu, jeho vizualizácie a manipuláciu s ním. Tento modul tiež obsahuje funkcie pre testovanie presnosti predikcie modelov.

Ôsma kapitola sa venuje samotnému porovnávaniu úspešnosti jednotlivých modelov, ktoré boli otestované na definovanom úseku komunikácie.

Deviata kapitola sa venuje návrhu, implementácii a popisu funkčnosti predikčného systému a jeho napojenia na existujúcu infraštruktúru.

Záverečná, desiata kapitola, sa venuje nastaveniu predikčného systému, jeho spusteniu a popisu potrebných závislostí.

1 DOJAZDOVÉ DOBY

Ako bolo uvedené, predikcia dojazdových dôb sa stáva čoraz dôležitejšou súčasťou dopravných systémov. Vzhľadom na skutočnosť, že tento typ predikcie často slúži vodičom vozidiel, ktorí sa tak môžu lepšie pripraviť na cestu a môžu predpokladanú dojazdovú dobu zarátať do svojich plánov, je potrebné aby bola predikcia čo najpresnejšia. V prípade výrazne (a postrehnuteľne) nepresne predikovanej hodnoty dojazdovej doby je potom prevádzkovateľ vystavený tlaku verejnosti.

Predikovanie dojazdových dôb sa používa na najrôznejších miestach, avšak jeho najväčšie využitie je na úsekoch diaľnic, ďalej je tiež využívaný v hromadnej mestskej doprave a v menšom merítku na úsekoch krátkeho rozsahu (napr. mestská premávka - informácie pre osobné automobily).

1.1 Zber dát pre dojazdové doby

Dáta pre dojazdové doby sa dajú získať rôznymi spôsobmi. Celkovo vedecké štúdie [1] spomínajú niekoľko metód, akými sa dá tento pravdivý udaj (ktorý však nie je k dispozícii v reálnom čase) zozbierať. Medzi ne patrí:

- Zmeranie dojazdovej doby pri reálnej jazde vo vozidle v daný čas, o ktorý sa zaujímame.
- Získanie informácii o pohybe vozidla pomocou GPS zariadenia (štúdia spomína prípade s autobusmi mestskej hromadnej dopravy).
- Dáta získané párovaním zachytených ŠPZ vozidiel a spočítanie času uplynutého medzi prvým a druhým záchytom.
- Dáta získané zo dvojitých smyčkových detektorov a následná identifikácia vozidiel podľa dĺžky, ktorú zachytili detektory zaznamenali. Potom sa znovu dopočíta čas medzi prvým a ďalším zachytením vozidla.
- Zachytávanie dát pomocou Bluetooth detektorov. Prebieha prostredníctvom párov detektorov na začiatku a konci sledovaného úseku, tieto detektory zachytávajú MAC adresy prechádzajúcich vozidiel a dojazdové doby sú vypočítané z rozdielu časov medzi zachyteniami rovnakej MAC adresy. Táto metóda má však jednu nevýhodu - má problémové využitie na miestach, kde sa kríži viacero komunikácií, pretože Bluetooth detektory zachytia nielen vozidlá, ktoré sa nachádzajú len na sledovanej komunikácii, ale aj na ostatných.

Všetky vyššie popísané metódy priamo merajú dojazdové doby, či už sledovaním vozidla (GPS) alebo pomocou dvojice detektorov (rozdiel času medzi prvým a ďalším zachytením).

Pri meraní dojazdových dôb je žiadúce tieto dojazdové doby merať na čo najkratších úsekoch. Preto sa dlhé úseky (nazvime ich **corridors**) delia na kratšie úseky (**links**). Toto delenie prebieha rozdelením - rezami (**slices**), na ktorých potom prebieha zber dát a samotné výpočty dojazdových dôb. [1]

1.2 Definície dojazdových dôb

Rozlišuje sa viacero definícií dojazdových dôb:

- pravdivá priemerná dojazdová doba
- priemerná dojazdová doba založená na príjazde
- priemerná dojazdová doba založená na odjazde

Tieto definície budú popísané a budú znázornené rozdiely medzi nimi. Praktickými rozdielmi pri výpočtoch podľa týchto definícií sa zaoberá podkapitola 1.3.

1.2.1 Pravdivá priemerná dojazdová doba

Pravdivá priemerná dojazdová doba $T_T(A)$ (true average travel time) je definovaná ako:

$$T_T(A) = \frac{\Delta x}{v(A)} = \Delta x \sum_{i=1}^{n} \frac{d_i}{T_i} \tag{1.1}$$

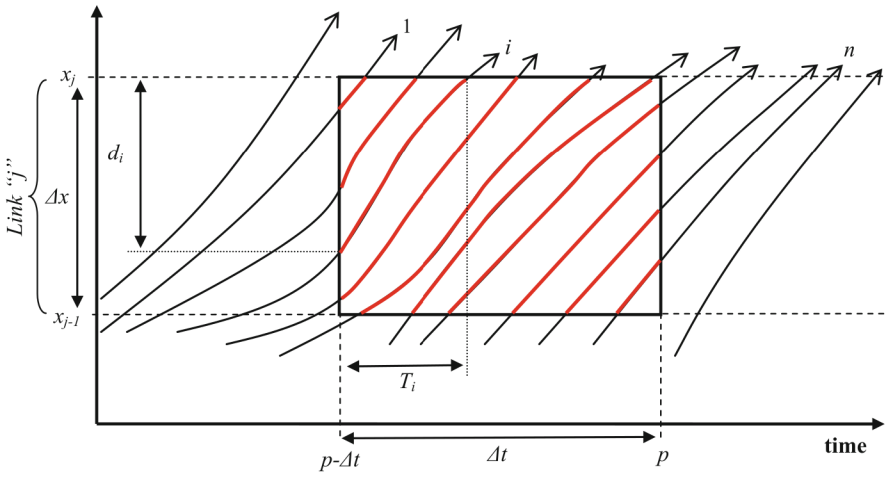
kde Δx je dĺžka úseku A (corridor), na ktorom je dojazdová doba sledovaná, $v(A)$ je priemerná rýchlosť na tomto úseku, d_i je vzdialenosť prejdená i-tym vozidlom v úseku A a T_i je čas strávený i-tym vozidlom v úseku A.

Definícia dojazdovej doby sa tiež viaže k časovému oknu $\mathcal{P}$ (ohraničené $p - \Delta t$ a p) s dĺžkou agregačného intervalu Δt, počas ktorého prebieha zber dát. Je možné, že sa pri nevhodne zvolenom Δt nepodarí $T_T(A)$ spočítať, keďže nie je možné zistiť pozíciu vozidiel, ktoré sa v časoch $p - \Delta t$ a p nachádzajú vo vnútri úseku A. Pre názornosť je definícia $T_T(A)$ zobrazená na obrázku 1.1. [1]

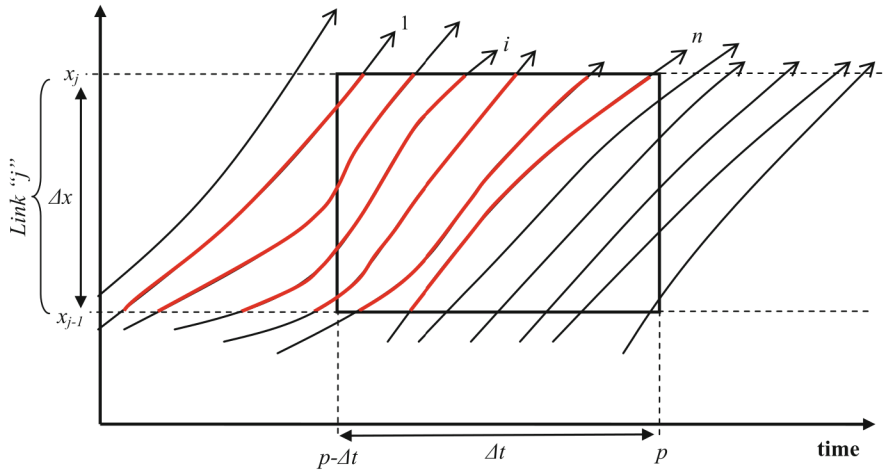
1.2.2 Priemerná dojazdová doba založená na príjazde

Priemerná dojazdová doba založená na príjazde $T_A(A)$ (arrival based average travel time) je definovaná ako priemerná dojazdová doba počas jazdy tých vozidiel, ktoré dosiahnu bod x_j v časovom okne P (obrázok 1.2).

Tento typ dojazdovej doby je získaný zo všetkých druhov priamych meraní založených na identifikácii vozidiel (popísané v podkapitole 1.1). [1]



Obr. 1.1: Znázornenie pravdivej priemernej dojazdovej doby [1]

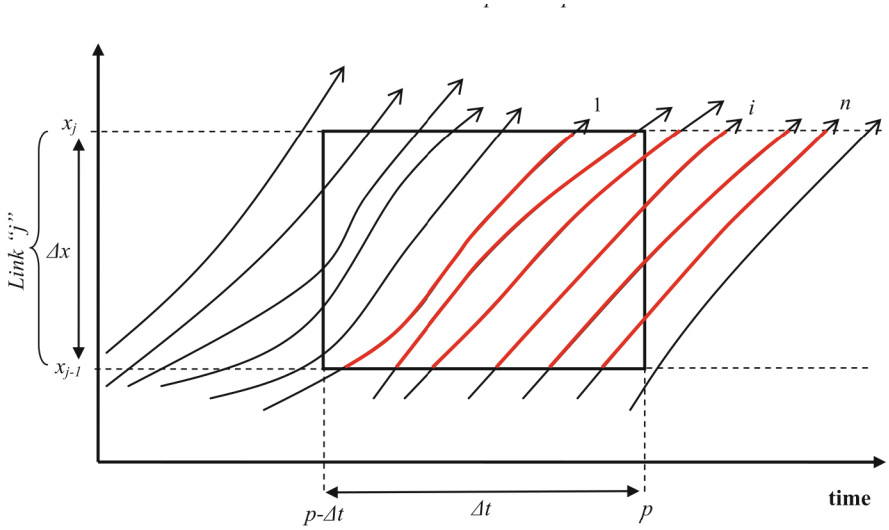


Obr. 1.2: Znázornenie dojazdovej doby založenej na príjazde [1]

1.2.3 Priemerná dojazdová doba založená na odjazde

Priemerná dojadová doba založená na odjazde $T_D(A)$ (departure based average trave time) je definovaná ako priemerná dojazdová doba vozidiel, ktoré vyrazili z bodu x_{j-1} v časovom okne P (obrázok 1.3).

Tento typ dojazdovej doby sa dá použiť len pri offline dopočítavaní dojazdových dôb, nakoľko vyžaduje informácie o budúcom príchode do bodu x_j. [1]



Obr. 1.3: Znázornenie dojazdovej doby založenej na odjazde [1]

1.3 Rozdiely definícií dojazdových dôb v praxi

Rozdiel medzi jednotlivými definíciami je v trajektóriach vozidiel, ktoré sú v nich obsiahnuté. Dojazdová doba $T_T(A)$ ako jediná obsahuje všetky trajektórie z daného časového okna P; $T_A(A)$ a $T_D(A)$ obsahujú podmnožinu týchto trajektórií. Rozdiel v priemernej dojazdovej dobe spočítanej podľa rôznych definícií potom závisí na relatívnom rozdiele medzi dojazdovou dobou a dĺžkou časového okna Δt, pričom čím väčšie budú časy dojazdových dôb v porovnaní s Δt, tým väčšie budú rozdiely v rámci skupiny vozidiel v jednom časovom okne naprieč definíciami.

Parameter Δt sa volí tak, aby nebol omnoho väčší než interval obnovy systému (napr. agregácia dát zo smyčkových senzorov) a výpočet dojazdovej doby bol primerane aktuálny a neboli vyfiltrované relevantné poklesy či nárasty dojazdovej doby. Pri výbere môžu nastať dve extrémne situácie [1], a to:

- $\Delta t >>$ **dojazdová doba**
  Dlhý agregačný interval spôsobí, že definície dojazdovej doby budú zdieľať majoritu trajektórií vozidiel, ktoré sa vyskytli v úseku v danom časovom okne P. Preto budú medzi hodnotami vypočítané podľa jednotlivých definícií minimálne rozdiely.
- $\Delta t <<$ **dojazdová doba**
  V tomto prípade, ktorý môže nastať napríklad zhustení premávky a pri kolonách, by sa mohlo stať, že jednotlivé definície nebudú zdieľať žiadne trajektórie vozidiel. V tom prípade môžu byť relatívne rozdiely medzi hodnotami vypočí-

tanými podľa rôznych definícií veľké.

Okrem tejto skutočnosti musí byť do úvahy zobraná aj skutočnosť, že vodičom vozidiel vchádzajúcich do úseku A v čase p je k dispozícii dojazdová doba z predchádzajúceho časového okna, ktorá je už nerelevantná. To implikuje požiadavku, aby bola dĺžka Δt časového okna P, čo najmenšia a zminimalizovalo sa oneskorenie vnesené vysokými hodnotami Δt.

1.4 Psychologické hľadisko predikcie dojazdových dôb a ich zobrazovania

Predchádzajúce podkapitoly sa zaoberali predovšetkým teóriou výpočtu referenčnej hodnoty dojazdovej doby (ktorá môže byť použitá pri porovnávaní a predikcii dojazdových dôb). Potrebné je ale spomenúť aj psychologické hľadisko predikcie dojazdovej doby - dva prípady nesprávnej predikcie.

Pre účely tejto práce si zadefinujeme termíny:

- **Dojazdová doba** automobilu z bodu A do bodu B je čas x, ktorý je potrebný na prekonanie vzdialenosti z bodu A do bodu B pri daných dopravných podmienkach v čase t.
- **Predikovaná dojazdová doba** je doba $\hat{x}$, predpovedaná v čase t, ktorú bude automobilu trvať presun z miesta A, kde sa nachádza v čase t, do miesta B, kde by sa mal nachádzať v čase $t+\hat{x}$.

Pri nesprávnej funkčnosti predikcii môžu nastať dve situácie, ktoré majú z psychologického hľadiska rôzny dopad:

- **Predikovaný čas je kratší ako v skutočnosti**
  Na základe zobrazenej informácie o dojazdovej dobe $\hat{x}$ v čase t si vodič vytvorí časový plán a predpokladá príjazd do destinácie v čase $t+\hat{x}$. Vzhľadom na to, že predikcia je nepresná, v skutočnosti príde do destinácie až v čase $t+\hat{x}+v$ (kde v je rozdiel medzi skutočnou a predikovanou dojazdovou dobou). Kvôli skutočnosti, že toto meškanie nepredpokladal, môžu nastať napr. hospodárske straty či iné nežiadúce javy a nesprávna predikcia je vnímaná veľmi negatívne.
- **Predikovaný čas je dlhší ako v skutočnosti**
  V tomto prípade vodič síce príde v čase $t+\hat{x}$-v, aj keď predpokladal, že príde v čase $t+\hat{x}$, avšak, tento prípad nie je vnímaný negatívne.

Pri optimalizácii predikčných algoritmov je teda dôležité sa dôrazne zameriavať na presnosť predikcie a hlavne na prvý popísaný prípad, ktorý verejnosť vníma veľmi negatívne.

2 STROJOVÉHO UČENIE PRE REGRESNÉ PROBLÉMY

Predikciou dojazdových dôb sa zaoberá veľké množstvo vedeckých prác. V týchto prácach boli na predikciu dojazdových dôb použité rôzne metódy založené na strojovom učení a štatistických metódach - od lineárnych modelov [2], SVR [3], ANN [4] až po kNN. V literatúre sa vyskytujú aj zmienky o modeloch postavených na takzvanom *ensemble learning* [5].

Tiež boli využité rôzne prístupy založené na data miningu [6] a veľa prác sa taktiež venuje predikcii založenej na rôznych implementáciách Kalmanovho filtra [7].

V tejto podkapitole budú stručne popísané niektoré z týchto metód.

2.1 Regresné algoritmy

Táto podkapitola sa venuje algoritmom, ktoré môžu byť využité ako regresné. Avšak v niektorých prípadoch boli nižie popísané algoritmy najprv predstavené ako klasifikačné, až neskôr boli od nich odvodené ich regresné verzie. Z toho dôvodu sú popísané obidve verzie algoritmov.

2.1.1 Lineárna regresia

Lineárna regresia je jeden zo základných algoritmov pre riešenie regresných problémov. Tento algoritmus predpokladá, že existuje lineárny vzťah medzi vstupom a výstupom, resp. že exituje lineárna závislosť, ktorá s dostatočnou presnosťou aproximuje skutočnú závislosť [8]. Model lineárnej regresie má formu

$$f(\mathbf{X}) = \beta_0 + \sum_{j=1}^{p} \mathbf{X_j} \beta_j \tag{2.1}$$

kde $\beta_j, j = 0 \dots p$ sú neznáme koeficienty, ktoré sú určené prostredníctvom minimalizácie chybovej funkcie - metóda najmenších štvorcov, kedy vyberáme koeficienty - parameter $\beta = (\beta_0, \beta_1, \dots \beta_p)^T$ ktorá je popísaná takto

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2 \tag{2.2}$$

Je dôležité podotknúť, že linearita modelu v tomto prípade neznamená linearitu vo vstupnej premennej $\mathbf{X}$, ale linearitu v parametri β. [8]

Metóda najmenších štvorcov má ale viacero nevýhod, pretože model získaný touto metódou má tendenciu príliš kopírovať dáta, na ktorých bol naučený a stráca

vlastnosť generalizovať. Túto problematiku rieši viacero algoritmov, medzi inými **Ridge regression** (podkapitola 2.1.2).

2.1.2 Ridge Regression

Tento algoritmus je odvodený od algoritmu lineárnej regresie (podkapitola 2.1.1), tiež hľadá model vo forme, ako je popísaný v rovnici 2.1. Avšak pri optimalizácii je do chybovej funkcie zakomponovaný člen, ktorý vyjadruje komplexnosť modelu.

Keďže modely s príliš vysokou komplexnosťou majú veľké hodnoty koeficientov β, sú príliš veľké hodnoty β penalizované. Koeficienty $\hat{\beta}$ sú nájdené algoritmom Ridge regression (rovnica 2.3). [8]

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{2.3}$$

kde λ vyjadruje mieru penalizácie veľkosti koeficientov a tým aj komplexnosť modelu.

Týmto spôsobom je možné získať model, ktorý je odolnejší voči šumu a má väčšiu schopnosť generalizovať ako pôvodný model lineárnej regresie.

2.1.3 Lasso

Na rozdiel od algoritmu Ridge, algoritmus Lasso [8] využíva pri regularizácii veľkosti koeficientov normu L_1 (viď rovnica 2.4). Navyše má schopnosť vybrať tie najsignifikantnejšie príznaky - koeficienty u ostatných prvkov príznakového vektora sú nahradené nulami.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{2.4}$$

2.1.4 ElasticNet

ElasticNet [9] je regresný algoritmus, ktorý v sebe spája regularizačné princípy algoritmov Lasso a Ridge. Využíva na penalizáciu normy L_1 aj L_2 a predpis pre nájdenie koeficientov je nasledovný:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1 \right\} \tag{2.5}$$

kde λ_1 a λ_2 kontrolujú veľkosť regularizácie podľa noriem L_1 (λ_1) a L_2 (λ_2).

2.1.5 Support Vector Regression

SVR (Support Vector Regression) je regresný algoritmus založený na algoritme SVM (Support Vector Machines). SVM je klasifikačný algoritmus, v pôvodnej verzii bol určený na binárnu klasifikáciu.

SVM

SVM je založený na myšlienke, pri ktorej dáta na binárnu klasifikáciu (vektory) transformujeme z ich pôvodného priestoru s dimenziou $\mathcal{D}_1$ do nového priestoru príznakov $\mathcal{D}_2$(feature space), ktorého dimenzia je väčšia, čiže $\mathbf{dim}(\mathcal{D}_\mathbf{1}) < \mathbf{dim}(\mathcal{D}_\mathbf{2})$. V tomto novom priestore je potom hľadaná nadrovina, ktorá optimálne rozdeľuje trénovacie dáta na dve skupiny. SVM nadrovinu vyberá tak, aby bol takzvaný margin - súčet vzdialeností medzi nadrovinou a najbližšími prvkami z obidvoch klasifikovaných tried najväčší možný. [10] Transformácia do vyššie-dimenzionálneho priestoru príznakov sa deje pomocou takzvaného kernel trick-u. Týmto je možné previesť lineárne neseparovateľnú úlohu na lineárne separovateľnú. [11]

SVR

Za účelom regresnej predikcie bol z SVM vytvorený algoritmus SVR. SVR je tiež založený na myšlienke kernel trick-u, v nadpriestore sa pokúša nájsť koeficienty funkcie, ktoré by spĺňali nasledujúci predpis uvedený v rovniciach 2.6:

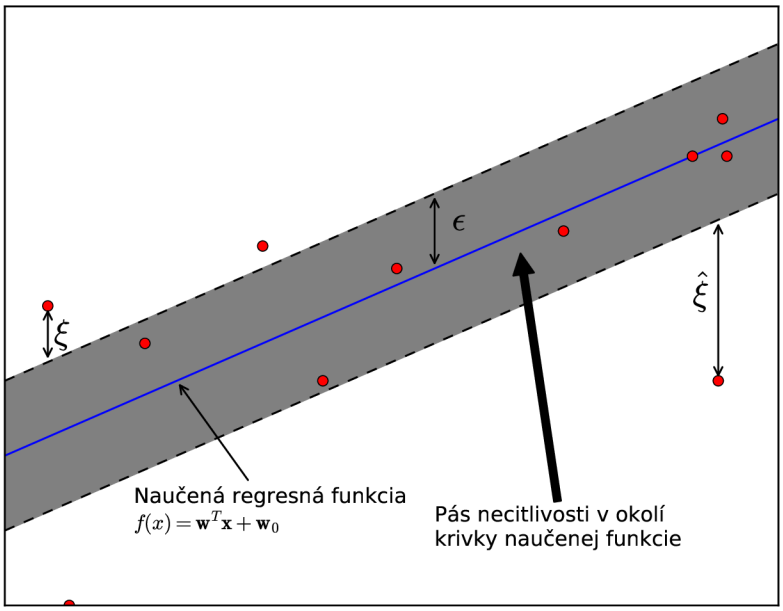
$$\begin{aligned} (\mathbf{w}^T\mathbf{x}_i + \mathbf{w}_0) - y_i &\leq \epsilon + \xi_i \qquad i = 1, \ldots, n \\ y_i - (\mathbf{w}^T\mathbf{x}_i + \mathbf{w}_0) &\leq \epsilon + \hat{\xi}_i \qquad i = 1, \ldots, n \\ \xi_i, \hat{\xi}_i &\geq 0 \qquad i = 1, \ldots, n \end{aligned} \tag{2.6}$$

Premenná ϵ udáva vzdialenosť od krivky naučenej funkcie, vrámci ktorej sa neuplatňuje chybová funkcia (takzvaný *pás necitlivosti*). Premenné ξ a $\hat{\xi}$ zase reprezentujú chybu za prekročenie pásu necitlivosti.

Učenie algoritmu prebieha hľadaním minimálnej hodnoty výrazu 2.7:

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}(\xi_i + \hat{\xi}_i) \tag{2.7}$$

kde $\mathbf{C}$ je regularizačný koeficient, ktorý kontroluje zložitosť modelu - resp. dôležitosť nepochybenia. Jeho vysoké hodnoty môžu spôsobiť preučenie algoritmu, ktorý potom stráca schopnosť generalizovať nové vzorky. Schematické znázornenie SVR je na obrázku 2.1.



Obr. 2.1: Znázornenie funkčnosti SVR

2.1.6 k-Nearest Neighbour

Algoritmus kNN - algoritmus k-najbližšieho suseda je pôvodne klasifikačný algoritmus prirodzene schopný klasifikácie do viacerých kategórií.

Funguje na princípe hľadania k-najbližších dátových vektorov (bodov) x_j, kde $j = 1 \ldots k$, k bodu z, pre ktorý chceme zistiť príslušnosť ku triede. Termín *najbližší* v tomto prípade vyjadruje blízkosť ako vzdialenosť podľa definovanej metriky medzi bodom z a bodmi x_j. Všeobecne sa využívajú euklidovská vzdialenosť (predpis 2.8), prípadne metrika manhattan (predpis 2.9).

$$vzdialenosť(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2} \tag{2.8}$$

$$vzdialenosť(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d}|x_i - y_i| \tag{2.9}$$

kde $\mathbf{x}$ a $\mathbf{y}$ sú vektory, a d je dimenzia týchto vektorov.

Po nájdení týchto *najbližších* vektorov je trieda $\hat{y_z}$ bodu z predikovaná ako trieda s najväčším zastúpením medzi triedami bodov x_j. [10]

Tento algoritmus má teda dva parametre - počet najbližších susedov, ktorý majú byť použitý pri predikcii k a metriku vzdialenosti, ktorými má byť posudzovaná *blízkosť*. Na rozdiel od výberu metriky, ktorá sa vo väčšine prípadov volí ako euklidov-

ská, je voľba parametra k náročnejšia hlavne v prípade dát, ktoré majú zložitejšiu štruktúru a sú vo veľkej miere ovplyvnené šumom.

Medzi výhody algoritmu kNN patrí jeho prirodzená schopnosť modelovania nelineárnych vzťahov medzi dátami a tiež výrazná odolnosť voči šumu, pokiaľ je správne nastavený parameter k. Z toho dôvodu bola vytvorená jeho regresná verzia.

Regresná verzia zdieľa fázu hľadania najbližších susedov k danému bodu s klasifikačnou verziou, ale po nájdení najbližších susedov sa predikovaná hodnota $\hat{y_z}$ vypočíta ako priemer výstupných hodnôt y_j, čiže

$$\hat{y_z} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{2.10}$$

kde k je počet najbližších susedov.

Ďalším možným spôsobom výpočtu výslednej hodnoty $\hat{y_z}$ regresnej funkcie je vážený priemer hodnôt y_j, pričom na bližšie body je kladený väčší dôraz ako na vzdialenejšie body, teda

$$\hat{y_z} = \frac{\sum_{j=1}^{k} y_j c_j}{\sum_{j=1}^{k} c_j} \tag{2.11}$$

Veľkosť koeficientov - váh pre jednotlivé susedné body môže byť určená automaticky s použitím funkcie - takzvaného *kernelu* ako generátoru týchto váh. Jedným z možných *kernelov* je Gaussova funkcia, ktorú zadefinujeme takto:

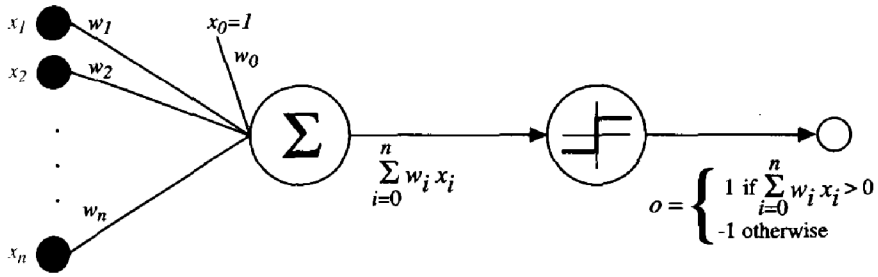
$$kernel = e^{-\frac{(\mathbf{z}-\mathbf{x_j})^2}{\lambda}} \tag{2.12}$$

kde $\mathbf{z}$ a $\mathbf{x_j}$ sú vektory a λ koeficient, ktorým sa kontroluje rýchlosť zmenšovania váh so vzdialenosťou bodu x_j od bodu z, pre ktorý chceme predikovať $\hat{y_z}$.

2.1.7 Neurónové siete

Neurónové siete sú ďalším algoritmom strojového učenia. Sú založené na princípe, akým sú poprepájané neuróny v ľudskom mozgu. Ich základnou jednotkou je perceptron (znázornenie na obrázku 2.2), ktorý je zjednodušeným matematickým modelom neurónu. Existuje viacero typov neurónových sietí (dopredné, so spätnou väzbou...), ktoré sa využívajú na klasifikačné aj regresné úlohy.

Jednoduchá neurónová sieť má 3 vrstvy - vstupnú (input layer), skrytú (hidden layer) a výstupnú (output layer). Rozmery vstupnej vrstvy závisia na dimenzii vstupných dát (resp. rovnajú sa), počet neurónov vo výstupnej vrstve závisí na aplikácii (pre regresiu zvyčajne jeden neurón, pre klasifikáciu viac). [8]



Obr. 2.2: Perceptron [12]

Neuróny môžu mať rôzne prenosové funkcie - sigmoidálnu, hyperbolickú tangentu, skokovú a iné [13]. Prehľad používaných prenosových funkcií je možné nájsť v [14].

Jednotlivé neuróny sú medzi sebou poprepájané, v prípade doprenej neurónovej siete sú v typicky prepojené všetky neuróny vo vrstve $\mathcal{L}_k$ na všetky neuróny vrstvy $\mathcal{L}_{k+1}$. Prepojenia nie sú obojsmerné, ale vždy len z vrstvy $\mathcal{L}_k$ na $\mathcal{L}_{k+1}$.

Na učenie neurónovej siete sa používajú rôzne algoritmy, pre doprednú neurónovú sieť algoritmus so spätným šírením chyby (error back-propagation), ktorý sa snaží o minimalizáciu chybovej funkcie, ktorá má pre regresné účely tvar uvedený rovnicou 2.13.

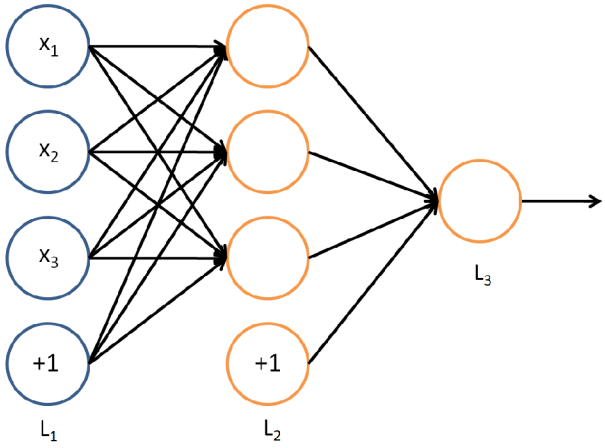
$$\mathcal{R}(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2 \tag{2.13}$$

kde K je počet neurónov vo výstupnej vrstve, N je počet príznakových vektorov v množine trénovacích dát, y_{ik} je očakávaná hodnota výstupného neurónu i pre k-ty trénovací vstup a $f(x_i)$ je hodnota výstupu k-teho neurónu vo výstupnej vrstve pre i-ty trénovací vektor. [8]

Najpoužívanejší typ neurónovej siete pre regresné úlohy je dopredná neurónová sieť (feed-forward neural network), znázornená na obrázku 2.3 ($\mathcal{L}_1$ je vstupná vrstva, $\mathcal{L}_3$ výstupná a $\mathcal{L}_2$ je skrytá vrstva). Pre regresiu sa doporučuje použiť vo vstupnej a výstupnej vrstve neuróny s lineárnymi prenosovými funkciami. [15]

2.2 Ensemble learning

V predchádzajúcich kapitolách boli popísané jednotlivé predikčné algoritmy. U týchto algoritmov ide vždy o to, aby sme našli nastavenie algoritmu, jeho hyperparametre, ktoré budú pre daný dataset najvhodnejšie a s ktorými bude mať model čo najvyššiu presnosť, resp. bude minimalizovať chybovú funkciu, ktorá bude pre tento účel definovaná.



Obr. 2.3: Dopredná neurónová sieť [16]

V niektorých prípadoch sa ale môže stať, že je viacero komplexných modelov, ktoré majú podobné hodnoty chýb a teda nie je jednoznačne možné určiť, ktorý je ten najlepší a najvhodnejší.

Potom je možné použiť takzvané **ensemble learning** - kombinovanie viacerých modelov a vytvorenie finálnej predikcie na základe predikcií jednotlivých modelov. [10]

2.2.1 Bagging

Jednou z metód ensemble learning je Bagging. Pozostáva z vytvorenia N modelov, z ktorých každý je trénovaný na vlastnom datasete $\mathbf{D}_i$, $i = 1 \ldots N$.

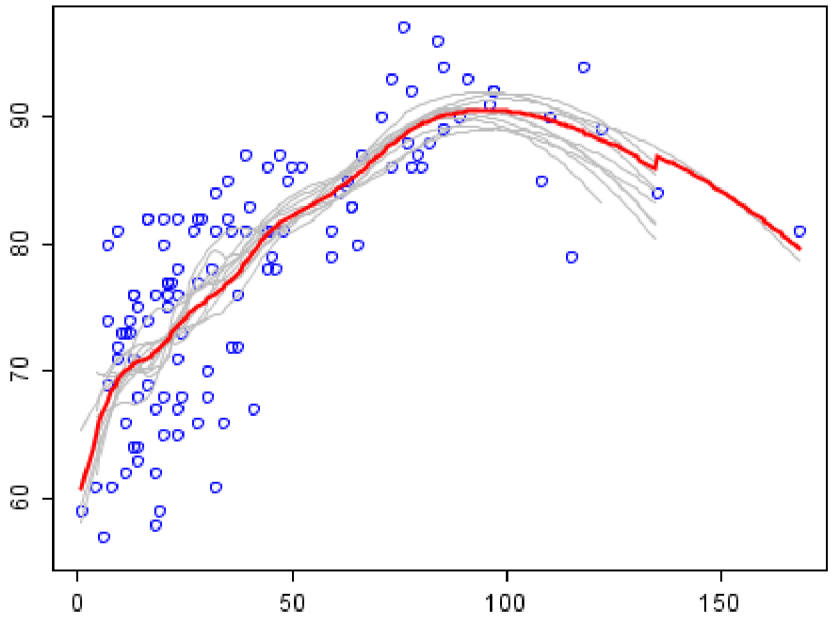
Datasety $\mathbf{D}_i \ldots \mathbf{D}_N$ sú vytvorené z pôvodného datasetu $\mathbf{D}$ podvzorkovaním takým spôsobom, že z datasetu $\mathbf{D}$[1] je pre každý dataset $\mathbf{D}_i$ náhodne vybraných M vzoriek. Tieto vzorky sú však ponechané aj v pôvodnom datasete a teda môže sa stať, že v niektoré vzorky sa nachádzajú vo viacerých datasetoch $\mathbf{D}_i \ldots \mathbf{D}_N$ a niektoré ani raz. [10]

Pri fáze predikcie je potom každému z N modelov predložený príznakový vektor, pre ktorý má byť predikovaná výstupná hodnota. Finálna predikcia je potom produktom agregácie predikcií jednotlivých modelov. Zlepšenie predikcie a jej stability v porovnaní s predikciami jednotlivých modelov je možné vidieť na obrázku 2.4.

2.2.2 Boosting

Boosting je ďalšia technika *ensemble learning*. Je založená na princípe kombinovania nepresných, slabých modelov za účelom zvýšenia ich presnosti tak, že postupne

[1] Datasety obsahujú vektory príznakov aj výstupných hodnôt



Obr. 2.4: Bagging - zlepšenie predikcie [17]

vytvára nové datasety $\mathbf{X}_i$ podvzorkovaním pôvodného datasetu $\mathbf{X}$ a na týchto datasetoch trénuje jednotlivé (*slabé*) modely. Na rozdiel od algoritmu Bagging (podkapitola 2.2.1) sú tieto datasety vytvárané deterministicky a nie náhodne.

Datasety sú vytvorené na základe výsledkov modelu z prechádzajúcej iterácie. Z datasetu sú potom vybrané vzorky, pri ktorých mal model z predchádzajúcej iterácie vysokú chybovosť, prípadne sú týmto vzorkám priradené vyššie váhy ako ostatným (to vyžaduje algoritmus, ktorý dokáže pracovať s rozličnými váhami na vzorkách) a vzorkám, ktoré boli predikované správne, sú váhy znížené. Pôvodným algoritmom predstavujúcim boosting je **AdaBoost**. [10]

2.3 Algoritmy data miningu

Pri rôznych úlohách, či už regresných alebo klasifikačných, je výhodné zistiť, ako sú dáta usporiadané, či existujú nejaké zhluky dát a podobne. Túto problematiku riešia algoritmy na báze takzvaného clusteringu. Vstupom týchto algoritmov sú iba dáta - vektory príznakov $\mathbf{X}$, pričom výstupom učenia je ich roztriedenie do zhlukov (clusterov). Jedná sa teda o učenie bez učiteľa - nie sú známe informácie o výstupe - či už trieda u klasifikácie alebo očakávaná hodnota u regresie.

Takto roztriedené dáta, ktoré formujú zhluky v príznakovom priestore môžeme potom použiť ako vstup pre učenie klasických regresných algoritmov. Tento prístup môže mať za následok zníženie chyby klasifikácie (regresie), pretože každý zhluk má svoju vlastnú inštanciu klasifikačného (regresného algoritmu). Jednotlivé inštancie budú použité na predikciu vtedy, keď nový bod, pre ktorý chceme predikovať, bude spadať do zhluku, ktorý bol použitý na trénovanie daného algoritmu.

V tejto podkapitole budú popísané niektoré používané clusterizačné algoritmy - od najjednoduchších a teda výpočetne najmenej náročných až po algoritmy, ktoré dokážu nájsť nekonvexné zhluky, ktoré sú však výpočetne omnoho náročnejšie.

2.3.1 K-Means

K-means je jeden z najpopulárnejších iteratívnych clusterizačných algoritmov. Jeho funkcia spočíva v rozdelení vstupnej množiny dát do K clusterov, pričom K je jeden zo vstupných parametrov algoritmu. Cluster je v tomto prípade množina bodov, ktorých vzájomné vzdialenosti sú malé v porovnaní so vzdialenosťami od bodov, ktoré neležia v clustre. Jednotlivé clustere sú reprezentované ich stredmi - vektormi μ_k, kde $k = 1 \dots K$. [18]

Cieľom K-Means je nájsť také rozdelenie vstupných bodov do clusterov r_{nk} a stredy týchto clusterov μ_k, aby bol súčet druhých mocnín vzdialeností bodov x_n od stredov clusterov, do ktorých vody patria - μ_k, čo najmenší (viď rovnica 2.14).

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{2.14}$$

Člen r_{nk} v rovnici 2.14 vyjadruje príslušnosť bodu x_n ku clusteru r_{nk}, čiže:

$$r_{nk} = \begin{cases} 1, & \text{ak } x_n \text{ je priradené ku clustru } k \\ 0, & \text{inak} \end{cases} \tag{2.15}$$

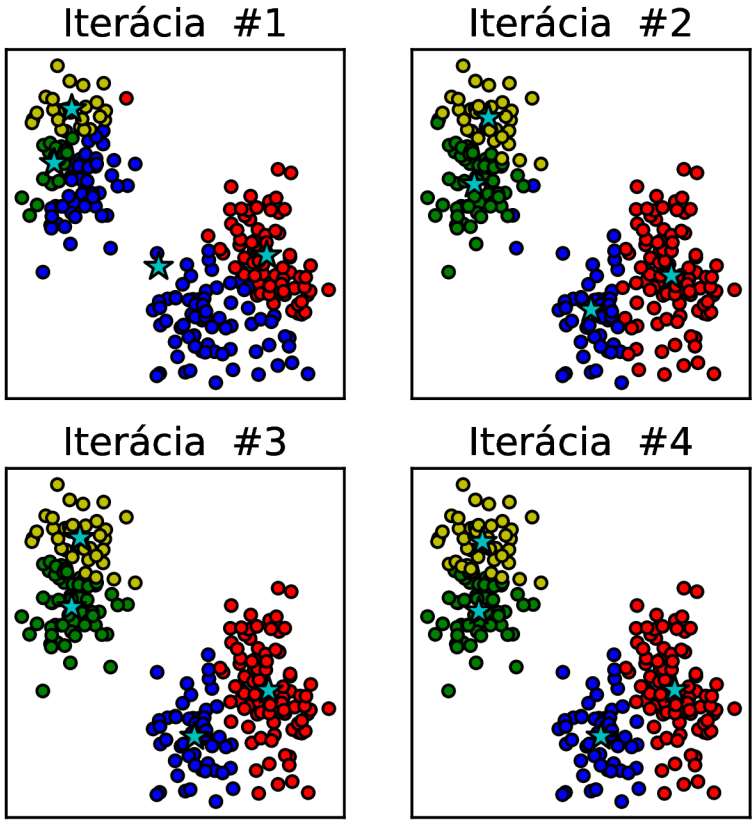
Beh algoritmu K-Means je nasledovný - po náhodnej inicializácii K clusterov prebiehajú dve fázy:

Fáza 1 Prepočet vzdialeností bodov x_n ku stredom clusterov μ_k a zostavenie matice príslušnosti bodov ku clusterom.

Fáza 2 Prepočítanie stredov clusterov μ_k.

Dĺžka behu je kontrolovaná konvergenciou minimalizácie rovnice 2.14.

Beh algoritmu K-Means je znázornený na obrázku 2.5. Body znázornené hviezdami predstavujú stredy clusterov - μ_k. Z priebehu jednotlivých iterácii je zrejmé, že K-Means postupne konverguje k jednému z možných optimálnych rozdelení bodov medzi clustre a minimalizuje výraz 2.14.



Obr. 2.5: Iterácie algoritmu K-Means

Hlavnou nevýhodou K-Means je, že dokáže priestor rozdeliť len do konvexných clusterov a nemôže teda zachytiť clustre, ktoré majú zložitejšiu obálku. [18]

Ďalšou nevýhodou je závislosť na počiatočných podmienkach, preto je vhodné nechať K-Means prebehnúť viac krát a nespoliehať sa na prvý výsledok.

Tiež je vhodné vyskúšať rôzne počty clusterov, pokiaľ nie je známy ich počet, ktorý by mal vzniknúť. Témou voľby vhodného počtu clusterov sa zaoberá veľké množstvo vedeckých prác.

2.3.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise - Clustering založený na hustote priestorých dát so šumom) je jedným z ďalších clusterizačných algoritmov. V porovnaní s K-Means sa jedná o pokročilý algoritmus, ktorý má však vyššiu výpočetnú náročnosť. Výhodou DBSCAN-u je možnosť vytvoriť a zachytiť štruktúru clusterov, ktoré majú nekonvexné obálky a značná odolnosť voči šumu. [19]

DBSCAN má však aj isté nevýhody - hlavným problémom je voľba použitej metriky, na ktorej je vo veľkej miere závislá kvalita clusterovania a tiež voľba parametrov algoritmu, ktoré by v ideálnom prípade mali byť určené expertom.

Algoritmus má dva parametre:

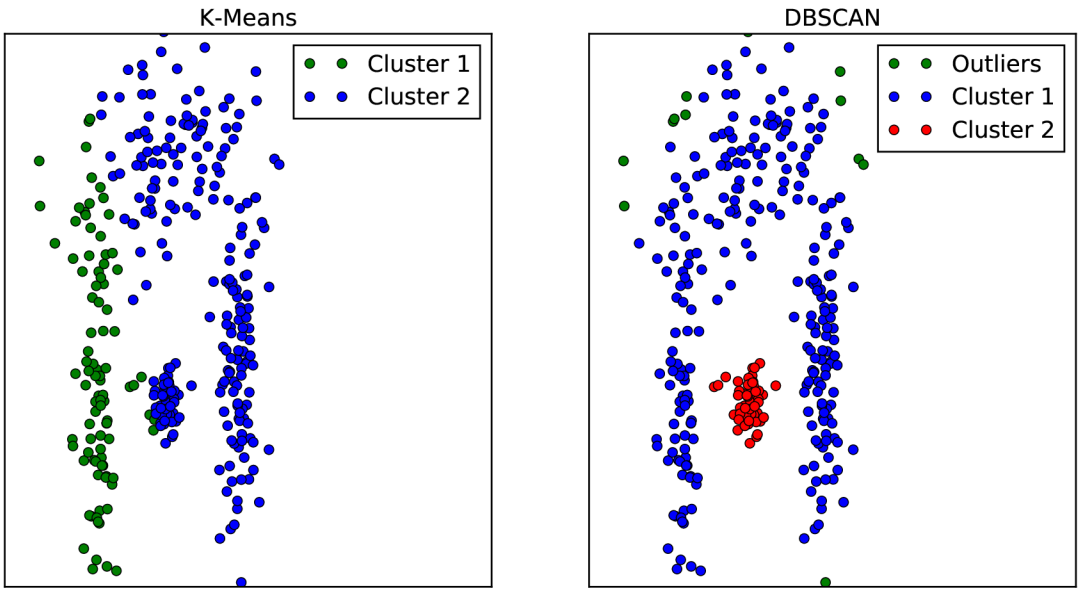
- ϵ - predstavuje maximálnu vzdialenosť bodu P_i od takzvaného *core* bodu C, ktorý reprezentuje cluster (v rámci clusteru je vždy minimálne jeden *core* bod), v rámci ktorej je bod P_i súčasťou clusteru.
- minPts - predstavuje najmenší potrebný počet bodov P_i v okolí bodu C v rámci vzdialenosti ϵ, aby bol bod C považovaný za *core* bod.

S využitím týchto dvoch parametrov potom DBSCAN body vyhodnocuje ako:

- ***Core* body** - Bod C, ktorý ma vo svojom v okolí v rámci vzdialenosti ϵ aspoň **minPts** bodov P_i. Tieto body P_i sú potom **priamo dosiahnuteľné** z *core* bodu C (body sú priamo dosiahnuteľné len z *core* bodov).
- **Dosiahnuteľné body** - Bod P je **dosiahnuteľný** z bodu M, pokiaľ existuje cesta naprieč bodmi $M_1 \dots M_n$, kde $M_n = P$, taká, že bod M_{n+1} je priamo dosiahnuteľný z bodu M_n (teda všetky body postupnosti musia byť *core* body).
- **Nezaraditeľné** (outliers) - Tieto body sú všetky body ktoré nie sú dosiahnuteľné z iných bodov.

Jednotlivé clustre sú potom formované ***core* bodmi**, ktoré sú navzájom priamo dosiahnuteľné a **dosiahnuteľnými bodmi** (ktoré nie sú *core* body, ale sú z *core* bodov dosiahnuteľné). [19]

Znázornenie a porovnanie vytvorenia clusterov algoritmami DBSCAN a K-Means je uvedené na obrázku 2.6. Je viditeľné, že zatiaľ čo K-Means umožnůje rozdelenie priestoru do clusterov konvexného tvaru, DBSCAN nájde de facto aj cluster obklopené inými clustrami a dokáže tak nájisť zložitejšie závislosti. Ako bolo naznačné v úvode podkapitoly, DBSCAN je necitlivý voči vzorkám dát, ktoré sa extrémne vymykajú štruktúre a usporiadaniu celku - označuje tieto body ako outliers - nezaraditeľné.



Obr. 2.6: Clusterizácia s využitím DBSCAN

2.4 Vyhodnocovacie kritériá

Pri hľadaní najvhodnejšieho modelu, resp. algoritmu, pre danú aplikáciu, je dôležité mať referenčné porovnávacie kritériá, ktoré sa môžu použiť na porovnanie rôznych prístupov. Táto podkapitola sa venuje najrozšírenejším kritériám, podľa ktorých sa dá zhodnotiť kvalita predikcie. Medzi takéto kritéria patrí napríklad MAE, MAPE, RMSE a RMSLE [20], ale tiež aj R^2 [21].

V tejto podkapitole je tiež popísaný konkrétny prístup vyhodnocovania kvality predikcie dojazdových dôb s prihliadnutím na negatívne stránky predikcie ako je popísaný v podkapitole 1.4 - metrika RMSLE.

Pre všetky metriky uvedené v tejto kapitole je jednotné značenie - $\hat{y}_i$ je predikovaná hodnota, y_i je skutočná hodnota v danom bode a N je počet vzoriek.

2.4.1 MAE

MAE (Mean Absolute Error), je metrika, ktorá vyjadruje ako ďaleko je v priemere predikovaná hodnota od skutočnej hodnoty. Je definovaná ako:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.16}$$

2.4.2 MAPE

Na rozdiel od MAE, **MAPE** vyjadruje relatívnu vzdialenosť predikovanej hodnoty od očakávanej, takže predstavuje metriku, ktorá sa dá ľahšie vyhodnotiť aj osobe, ktorá nie je doménový expert, resp. nie je jej známa štruktúra dát. Je definovaná ako:

$$\text{MAPE} = 100\frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i} \tag{2.17}$$

2.4.3 RMSE

Metrika **RMSE** vyjadruje, podobne ako MAE, priemernú chybu predikcie, avšak na rozdiel od MAE toleruje malé odchýlky od očakávanej hodnoty a výrazne penalizuje veľké. Definovaná je ako:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.18}$$

2.4.4 RMSLE

Na rozdiel od predchádzajúcich metrík, u ktorých nezáležalo na tom, či bola predikovaná hodnota nižšia alebo vyššia ako skutočná, táto metrika nie je symetrická. **RMSLE** penalizuje podhodnotenie skutočnej hodnoty viac ako nadhodnotenie. Táto metrika je definovaná ako:

$$\text{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\ln(\hat{y}_i + 1) - \ln(y_i + 1))^2} \tag{2.19}$$

2.4.5 R^2

R^2 je štatistická metrika, tiež nazývaná *coefficient of determination*. Vyjadruje množstvo variancie (rozptylu) obsiahnutého v závislej premennej, ktorý môže byť vysvetlený nezávislou (vstupnou) premennou. Jeho výpočet je možný podľa predpisu 2.20.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i)^2} \tag{2.20}$$

3 METÓDY PREDIKCIE DOJAZDOVÝCH DÔB

Táto kapitola sa zaoberá výskumnými prácami a projektami, ktorých tematikou je predikcia dojazdových dôb. Tieto práce je možné rozdeliť na dve skupiny - práce, ktoré sa zaoberajú jedným konkrétnym algoritmom a práce zaoberajúce sa porovnaním kvality predikcie rôznych typov algoritmov, prípadne príznakov.

Táto kapitola sa bude vo zvýšenej miere venovať hlavne druhej skupine, pretože poskytuje údaje, ktoré sú dôležité pri výbere správneho algoritmu na aplikáciu, ktorej sa venuje táto práca.

Vybrané štúdie budú rozobrané a budú uvedené metodiky, s akými sa postupovalo pri výbere príznakov a základe autormi zvolených metrík budú porovnané algoritmy, ktoré boli testované v rámci štúdií.

3.1 Qiao, Haghani, Hamedi - 2013 [22]

Autori v tejto štúdii popisujú porovnávanie modelov ARIMA, Kalmanovho filtra a kNN pre krátkodobú predikciu dojazdových dôb založenú na dátach z Bluetooth detektorov (zber dát pomocou Bluetooth detektorov je popísaný v podkapitole 1.1).

Táto práca tiež predstavuje modifikovanú metódu kNN, takzvanú kNN-T. Táto metóda využíva opakujúce sa vzory v dátach k vylepšeniu predikcie dojazdovej doby. Okrem základného konceptu kNN je použitý aj model, ktorý zohľadňuje trendovú zložku a hľadá podobnosti s trendom dát použitých ako vstup predikcie a trendom dát, ktoré boli použité pri trénovaní algoritmu.

Uvedené metódy sú v práci otestované a ako hlavná metrika kvality predikcie je použité kritérium MAPE (popísané v podkapitole 2.4.2). Ako najmenej chybové metódy sa ukázali kNN a kNN-T. Pri predikovaní dojazdovej doby pre hodiny najrušnejšie premávky, resp. špičiek, bola chyba MAPE 11.3% pre kNN a 10.7% pre kNN-T. Pri predikovaní celého dňa boli chyby 5.1% a 4.9%. V prípade špičiek bola chyba pre model ARIMA a model založený na Kalmanovom filtre o 2% resp. 3% vyššie ako u kNN.

Práca explicitne nezmieňuje žiadne príznaky, ktoré využila pri trénovaní prediktorov, zmieňuje sa však o priemernej rýchlosti a dojazdovej dobe, ktorá sa dá z bluetooth dát získať.

V práci je taktiež spomenutý fakt, že pri predikcii prichádza k oneskoreniu predikovanej hodnoty oproti skutočnej hodnote a táto skutočnosť sa týka všetkých typov prediktorov.

3.2 Lunde, Wolff - 2015 [5]

Táto práca poskytuje porovnanie dvoch rôznych prístupov k predikcii dojazdových dôb. Prvým z nich je takzvané *online learning* - učenie algoritmu, pri ktorom sú algoritmu predkladané vzorky dát, z ktorých sa tento algoritmus učí, avšak pri učení nie je potrebné predložiť vždy všetky vzorky naraz. Algoritmu stačí predložiť nové vzorky, ktoré následne zapracuje do modelu. Toto riešenie teda netrpí problémami týkajúcich sa času potrebného na učenie, keďže sa pri tréningu dáta vkladajú inkrementálne a nie celý použitý dataset naraz. Vďaka inkrementálnemu prísunu dát sa tiež model môže prispôsobovať štruktúre dát v čase.

Ďalšou popisovanou metodikou je takzvané *ensemble learning*. Pri tejto metodike je predikcia produktom kombinácie predikcií od viacerých modelov/algoritmov. Tento prístup potom umožňuje dosiahnuť vyššiu presnosť.

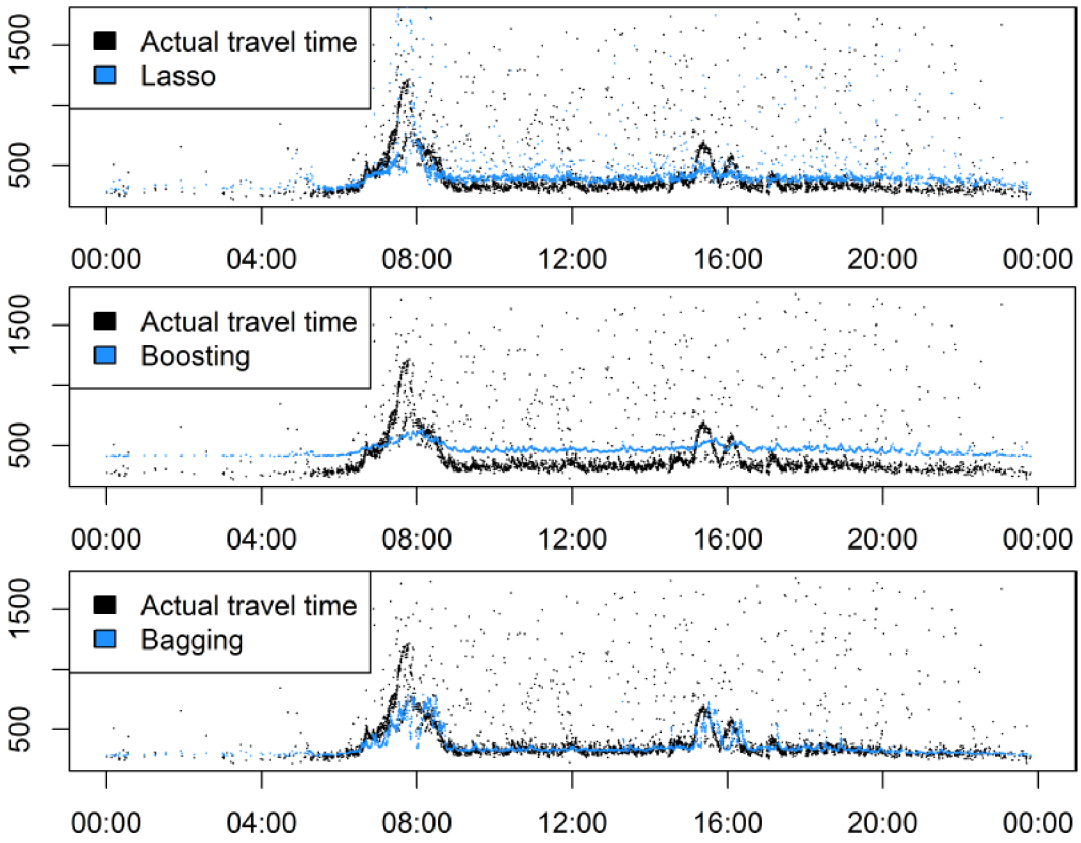
Autori porovnávali viacero algoritmov, spomedzi predstaviteľov *ensemble learning* vybrali algoritmy bagging, boosting, lasso a tiež riešenie založené na fuzzy systéme. Z pomedzi predstaviteľov *online learning-u* bola porovnávaná úspešnosť predikcie Kalmanovho filtra a algoritmu LOKRR.

Spomedzi algorimov je zaujímavý LOKRR, ktorý funguje na princípe rozdelenia datasetu podľa časovej domény, autori spomínajú delenie na časové okná s veľkosťou v ráde jednotiek minút. Pre jednotlivé časové okná sú potom vytvorené individuálne modely založené na KRR (Kernel Ridge Regression). Treba podotknúť, že keďže sa priebeh hodnoty dojazdovej doby mení a nie je zaručené, že jej krivka je vždy taká istá, je potrebné na vytvorenie modelu pre časové okno W_i použiť aj dáta z okien W_{i-1} a W_{i+1}.

K porovnaniu modelov bol využitý dataset zo 4.6 km dlhého úseku (dáta z úseku sa skladajú z meraní na vstupe a na výstupe); vstupnými parametrami modelov boli priemerná dojazdová doba a priemerný tok vozidiel na vstupe úseku za posledných 5 minút, pričom výstupom bola očakávaná hodnota dojazdovej doby.

Pre porovnanie modelov boli použité metriky MAE (podkapitola 2.4.1) a RMSE (podkapitola 2.4.3), zapojený bol tiež aj Friedmanov test [23]. Autori práce konštatujú, že zo získaných výsledkov nie je možné posúdiť, ktorý model je lepší a z výsledných priebehov reálnych a predikovaných dojazdových dôb je patrné, že v rôznych situáciach sa prejavujú silné a slabé stránky iných modelov. Žiadny teda nemá konzistentne vyššiu presnosť predikcie ako ostatné. Tieto zistenia platia pre obe metódy - *online learning* aj *ensemble learning*.

V práci je ale tiež spomenuté, že predstaviteľ *ensemble learning* - boosting vykazuje najmenšiu presnosť a pre predikciu (minimálne na testovacom datasete) nie je príliš vhodný. Porovnanie je ukázané na obrázku 3.1. Presnejšie predikcie naopak dosahujú algoritmy lasso a bagging.



Obr. 3.1: Porovnanie predikcie dojazdových dôb - lasso, boosting a bagging z [5]

Z práce tiež vyplýva, že je veľmi dôležité dáta, pred použitím na tréning prediktorov spracovať, napríklad odstránením dát, ktoré sa veľmi vymykajú štruktúre a usporiadaniu celku. Účinnosť tohoto opatrenia je viditeľná aj porovnaním kvality predikcie metód *ensemble learning*, u ktorých boli dáta predspracované a ktoré vykazujú lepšiu prediktívnu schopnosť, a *online learning*, u ktorých dáta predspracované neboli a ich kvalita predikcie podľa metrík RMSE a MAE je niekoľko krát horšia ako u *ensemble learning* metód.

3.3 Yildirimoglu, Geroliminis - 2013 [24]

Táto práca sa venuje predikcii dojazdových dôb založenej na využití teoretických znalostí a zákonitostí týkajúcich sa dopravného inžinierstva. Autori v práci nepredikujú dojazdovú dobu na celej komunikácia naraz (predikcia dlhého úseku - corridor) L, ale pre množstvo menších úsekov (links) $L_{sub,i}$, $i = 1 \dots N$, kde N je počet týchto úsekov vo celom úseku (terminológia je definovaná v podkapitole 1.1). Úseky sú fy-

zicky ohraničené detektormi (vzdialenosť detektorov pri komunikácii vyšetrovanej v štúdii je cca 500m).

Okrem priestorového delenia je dataset rozdelený aj časovo. Autori rozdelili dáta z každého úseku $L_{sub,i}$ tak, že každý deň je rozdelený na 180 časových okien. Každý deň je teda reprezentovaný $180xN$ premennými.

Navyše, po časovom a priestorovom rozdelení dát na okná autori využívajú clusterizačný algoritmus GMM (Gaussian Mixture Models), ktorý nájde zhluky podobných dát - k; v tomto prípade sa autori snažia nájisť zhluky dní s podobnými charakteristikami - rýchlostnými profilmi v danom úseku v daný čas. Toto však vzhľadom na počet premenných (dimenzionalitu dát, $dim = 180xN$) nie je možné, preto je predtým na celý dataset aplikovaný algoritmus PCA, ktorý nájde najvýznamnejšie principálne komponenty vo vstupnom datasete. Z týchto je potom zkonštruovaný nový dataset, pričom je vybraných len niekoľko prvých principálnych komponentov.

Ďalej autori postupovali vytvorením takzvanej *mapy zahltenia* (congestion map, viď obrázok 3.2). Táto mapa určuje pravdepodobnosť, že v clusteri k, segmente i, čase t a dni m bola pozorovaná kolóna (viď rovnica 3.1). Funkcia f v tejto rovnici, ktorá je implementovaná ako algoritmus na detekciu kolón (detekuje rozdiely v rýchlosti na vstupe a výstupe úseku) priraďuje jednotlivým úsekom i v čase t a dni m hodnoty ako je ukázané v rovnici 3.2.

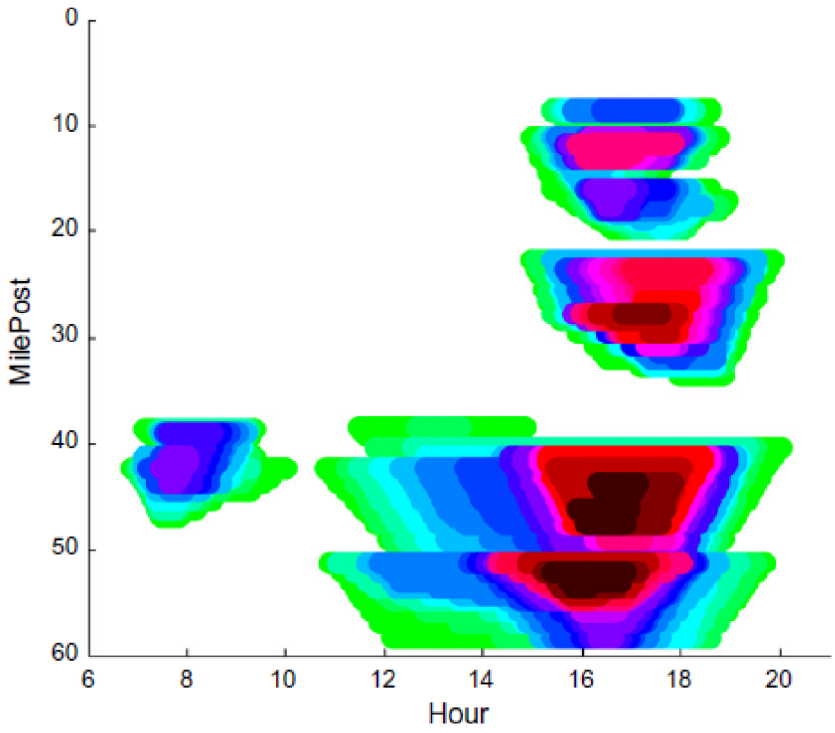
$$p^k(i,t) = \frac{1}{D_k} \sum_{m=1}^{D_k} f(i,t,m) \tag{3.1}$$

$$f(i,t,m) = \begin{cases} 1, & \text{ak je segment } i \text{ zahltený v čase } t \text{ v dni } m \\ 0, & \text{inak} \end{cases} \tag{3.2}$$

V tejto práci autori vypočítavajú dojazdovú dobu na základe priemernej rýchlosti úsekoch - rýchlostných profilov. Pre nezahltené a zahltené podmienky sú rýchlostné profily vytvorené zvlášť.

Predikcia potom prebieha tak, že historická mapa zahltenia je porovnávaná s informáciami (zahltené/nezahltené) o aktuálnom zahltení. Pri tomto porovnávaní je vybraný blok na mape zahltenia, ktorého tvar je najbližší aktuálnym informáciám a na základe tohto je vybraný rýchlostný profil, ktorý najviac zodpovedá prítomnosti. Z rýchlostného profilu a veľkosti úsekov (segmentov) i je potom vypočítaná očakávaná dojazdová doba.

Autori v práci použili na popísanie kvality predikcie metriky MAE a MAPE, ich prístup dosiahol MAPE $1 - 4\%$. V porovnaním s predpoveďou dojazdovej doby na základe hodnoty dojazdovej doby vypočítanej z predchádzajúcich prejazdov vozidiel dosahoval v závislosti na podmienkach zlepšenie o $0 - 50\%$.



Obr. 3.2: Príklad mapy zahltenia z [24]

3.4 Elhenawy, Chen, Rakha - 2014 [6]

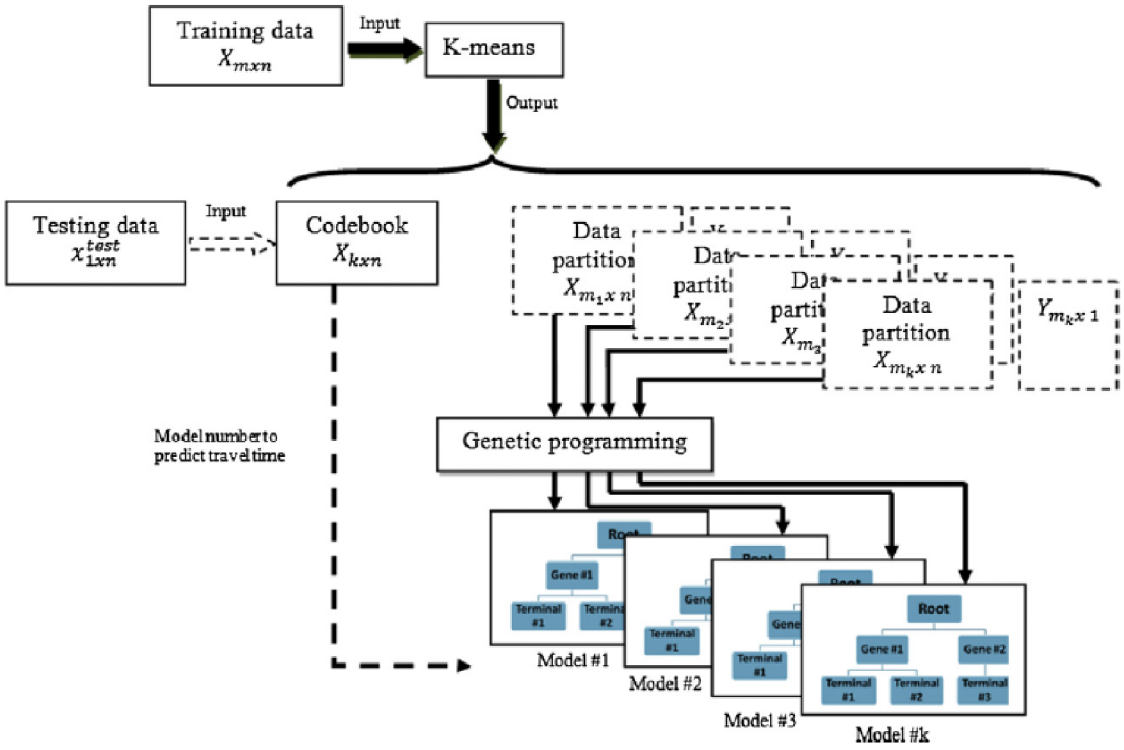
Autori tejto práce využili v porovnaní s ostatnými prácami iný prístup k predikcii dojazdových dôb a to clusterizačné algoritmy a genetické programovanie.

Ich novátorský prístup spočíva nasledujúcich krokoch:

1. Na pôvodný dataset $\mathbf{X}$ s dimenziou mxn je aplikovaný clusterizačný algoritmus K-Means (podkapitola 2.3.1), ktorý rozdelí dataset na K zhlukov - clustre $\mathbf{X}_i$ s dimenziami m_ixn, kde $i = 1 \dots K$
2. Z každého clustru $\mathbf{X}_i$ sú vytvorených nových S sub-datasetov $\mathbf{B}_{ij}$, kde $j = 1 \dots S$, získavame tak SxK sub-datasetov.
3. Pre každý z sub-datasetov $\mathbf{B}_{ij}$ bude vytvorený model. Vytvorenie modelu prebieha optimalizáciou jeho koeficientov pomocou genetického programovania - efektívnej optimalizačnej techniky založenej na biologickom princípe. Finálny model pre cluster $\mathbf{X}_i$ je potom lineárnou kombináciou modelov naučených na datasetoch $\mathbf{B}_{ij}$, kde $j = 1 \dots S$. Váhy pre lineárnu kombináciu sú nájdené metódou najmenších štvorcov.

Tvorba sub-datasetov v kroku 2. vyplýva z princípu podvzorkovania algoritmom Bagging (podkapitola 2.2.1). S využitím princípu tejto metódy sú potom trénované modely v kroku 3, pre každý dataset $\mathbf{B}_{ij}$ a tiež sú agregované ich predikcie. Štruktúra

celého modelu je znázornená na obrázku 3.3.



Obr. 3.3: Štruktúra predikčného modelu uvedeného v [22]

Štúdia bola vykonaná na úseku dlhom 60 km, ktorý je rozdelený na 58 sekcií. Uvádza, že ako premenné na predikovanie dojazdovej doby boli využité dáta o priemernej rýchlosti na jednotlivých sekciách agregované do 5 minútových intervalov, teda matica vstupných premenných sa skladá z priemerných rýchlostí v daných sekciách v daný časový interval.

Autori v štúdii využili na vyhodnotenie metriku MAE a MAPE. Konštatujú tiež, že proti modelu dojazdových dôb založenom na posledných známych hodnotách dojazdových dôb má nimi navrhnutý model o 76% menšiu chybu.

4 ZDROJE DÁT

Modely na predikciu dojazdových dôb na jednotlivých definovaných úsekoch sú vytvorené na základe dát dodaných firmou CAMEA. Tieto dáta sú získané s využitím bluetooth detektorov (párovanie MAC adries zariadení) a tiež párovaním zachytených ŠPZ (popisuje podkapitola 1.1) - tieto identifikátory sú anonymizované. Okrem toho sú tiež k dispozícii dáta o počte vozidiel, obsadenosti vozovky a rýchlosti, ktoré sú získané z viacerých typov snímačov.

Dostupné sú dáta z troch zdrojov:

- senzorické zo senzorov, ktoré sa nachádzajú na úseku
- dojazdové doby v čase t spočítané na základe dát dostupných v čase t, takzvaná okamžitá dojazdová doba
- dojazdové doby spätne dopočítané s využitím všetkých zaznamenaných dát - reprezentujú správnu hodnotu dojazdovej doby

Dáta zo všetkých troch zdrojov zdieľajú identifikátor, ktorým je čas, teda premenná **calculation_time_local** , ktorá sprostredkúva informácie o čase záznamu dát v lokálnom časovom pásme.

V nasledujúcich podkapitolách budú jednotlivé zdroje dát priblížené.

4.1 Senzorické dáta

Senzorické dáta sú spravidla agregované po 5-tich minútach a môžu pochádzať zo senzorov viacerých druhov. Patria medzi ne:

- **det1** - radarové detektory, poskytujú údaje o priemernej rýchlosti vozidiel, obsadenosti vozovky a počte vozidiel, ktoré detektor zaznamenal za agregačný interval
- **det2** - detektory na báze indukčnej smyčky - poskytujú údaje o celkovom počte vozidiel a počte vozidiel rôznych druhov (autobus, osobný automobil...) za agregačný interval

Dáta z týchto sú popísané v nasledujúcich paragrafoch.

4.1.1 det1

Najpoužívanejším druhom senzorov sú senzory typu **det1**, dáta z ktorých boli využité pri tvorbe predikčných modelov.

det1

Jedná sa o položku popisujúcu umiestnenie senzora, teda je to identifikátor senzora, ktorý vyjadruje jeho príslušnosť k rezu.

det1_velocity

Premenná reprezentujúca medián rýchlostí vozidiel v jazdnom pruhu, ktorý sníma senzor, za dobu agregačného intervalu.

det1_occupancy

Reprezentuje obsadenosť pruhu, ktorý sníma senzor, za dobu agregačného intervalu. Obsadenosť vyjadruje mieru zaplnenie jazdného pruhu.

det1_count

Počet vozidiel, ktorý bol zaznamenaný senzorom za dobu agregačného intervalu.

4.1.2 det2

Ďalším rozšíreným typom senzorov sú detektory využívajúce princíp snímania prostredníctvom indukčných smyčiek.

det2

Podobne ako pri **det1**, aj tu sa jedná o popis umiestnenia senzora.

det2_count

Premenná reprezentujúca počet zachytených vozidiel za dobu agregačného intervalu prostredníctvom indukčnej smyčky.

4.1.3 Okamžité dojazdové doby

Okamžité dojazdové doby - v dátach označené ako **bck_prediction_tt** - patria tiež medzi dodané dáta. Ako je už spomenuté, sú spočítané prostredníctvom párovania rovnakých ŠPZ vozidiel, prípadne MAC adries bluetooth zariadení nachádzajúcich vo vozidlách.

Dostupné dáta sú výstupom momentálne používaného modelu firmy CAMEA na predikciu dojazdových dôb založenom na okamžitých dojazdových dobách vozidiel v danom úseku za poslednú minútu. Jedná sa teda o priemernú dojazdovú dobu

založenú na príjazde (podkapitola 1.2.2). Vstupy do tohto modelu tiež podliehajú filtrácii mediánom, ktorá odstráni extrémne hodnoty.

4.1.4 Spätne dopočítané dojazdové doby

Spätne dopočítané dojazdové doby - **tt_real** sú spočítané offline ako dojazdové doby založené na odjazde (podkapitola 1.2.3), predstavujú teda pravú hodnotu dojazdovej doby vozidla, ktoré v čase t vyrazilo z miesta x_j a malo cieľovú destináciu x_{j+1}, do ktorej dorazilo v čase $t + T_D$. Spočítané sú priemerné dojazdové doby, ktoré mali vozidlá v danej minúte.

Dáta spomenuté v podkapitole 4.1.3 - okamžité dojazdové doby - teda v ideálnom prípade presne kopírujú dáta opísané v tejto podkapitole. Kvôli veľkým dojazdovým dobám a relatívne malej veľkosti okna (malá veľkosť časového okna je dôležitá pre malé oneskorenie systému a jeho frekvenciu obnovovania) sú však časovo oneskorené.

4.2 Chybné dáta

V *surových* dátach získaných reálnych podmienok sa môžu nachádzať aj chybné dáta. Tieto sa nachádzajú aj v analyzovaných dátach z firmy CAMEA, kde sú položky, ktoré majú neplatné hodnoty vyjadrené symbolom "-1". Táto skutočnosť je jasne viditeľná na zobrazených boxplot-och, kde sú tieto hodnoty vidieť ako minimá. Je dôležité o týchto chybných dátach vedieť a počítať s nimi pri ďalšom spracovaní.

5 ANALÝZA REÁLNYCH DÁT

Pre prvotnú analýzu boli dodané dáta z časového intervalu 16.10.2016 - 13.2.2017 z bližšie nedefinovaného úseku. Je teda potrebné podotknúť, že tieto dáta nemusia byť reprezentatívnou vzorkou dát zo všetkých sekcií, na ktorých sú predikované dojazdové doby.

Táto skutočnosť je však braná do úvahy aj v prácach rozobraných v kapitole 3, kde boli vo väčšine analyzované jednotlivé úseky samostatne a nie ako celok, keďže každý úsek môže byť niečím špecifický.

5.1 Analýza dát zo senzorov

Analyzovaný úsek je zložený z rezov (slice-ov) označených ako slice1, slice2, slice3, slice4, slice5, slice6 a slice7.

Využité sú na ňom senzory typu **det1** a **det2**, pričom na každom úseku sa nenachádzajú všetky druhy senzorov. Na nasledujúcich grafoch je znázornené rozloženie hodnôt premenných.

Z dôvodu prehľadnosti a využiteľnosti sú dáta z jednotlivých senzorov toho istého typu patriacich pod jeden slice zlúčené (agregované). Agregácia prebiehala špecificky pre každú premenné v závislosti na type veličiny a teda spôsob agregácie bude popísaný pre každú premennú zvlášť.

5.1.1 det1

Tieto senzory boli dostupné na rezoch slice1, slice2, slice3, slice4, slice5.

det1_velocity

Rozloženie hodnôt premennej **det1_velocity** je ukázané na obrázku 5.1. Agregácia dát z jednotlivých senzorov patriacich k jednému senzorovému rezu (slice-u) prebiehala ako aritmetický priemer hodnôt.

det1_occupancy

Rozloženie hodnôt premennej **det1_occupancy** je ukázané na obrázku 5.2. Agregácia dát z jednotlivých senzorov patriacich k jednému senzorovému rezu (slice-u) prebiehala ako súčet jednotlivých hodnôt.

det1_count

Rozloženie hodnôt premennej **det1_count** je ukázané na obrázku 5.3. Agregácia dát z jednotlivých senzorov patriacich k jednému senzorovému rezu (slice-u) prebiehala ako súčet jednotlivých hodnôt.

5.1.2 det2

Tieto senzory boli dostupné na rezoch slice4, slice5, slice6 a slice7.

det2_count

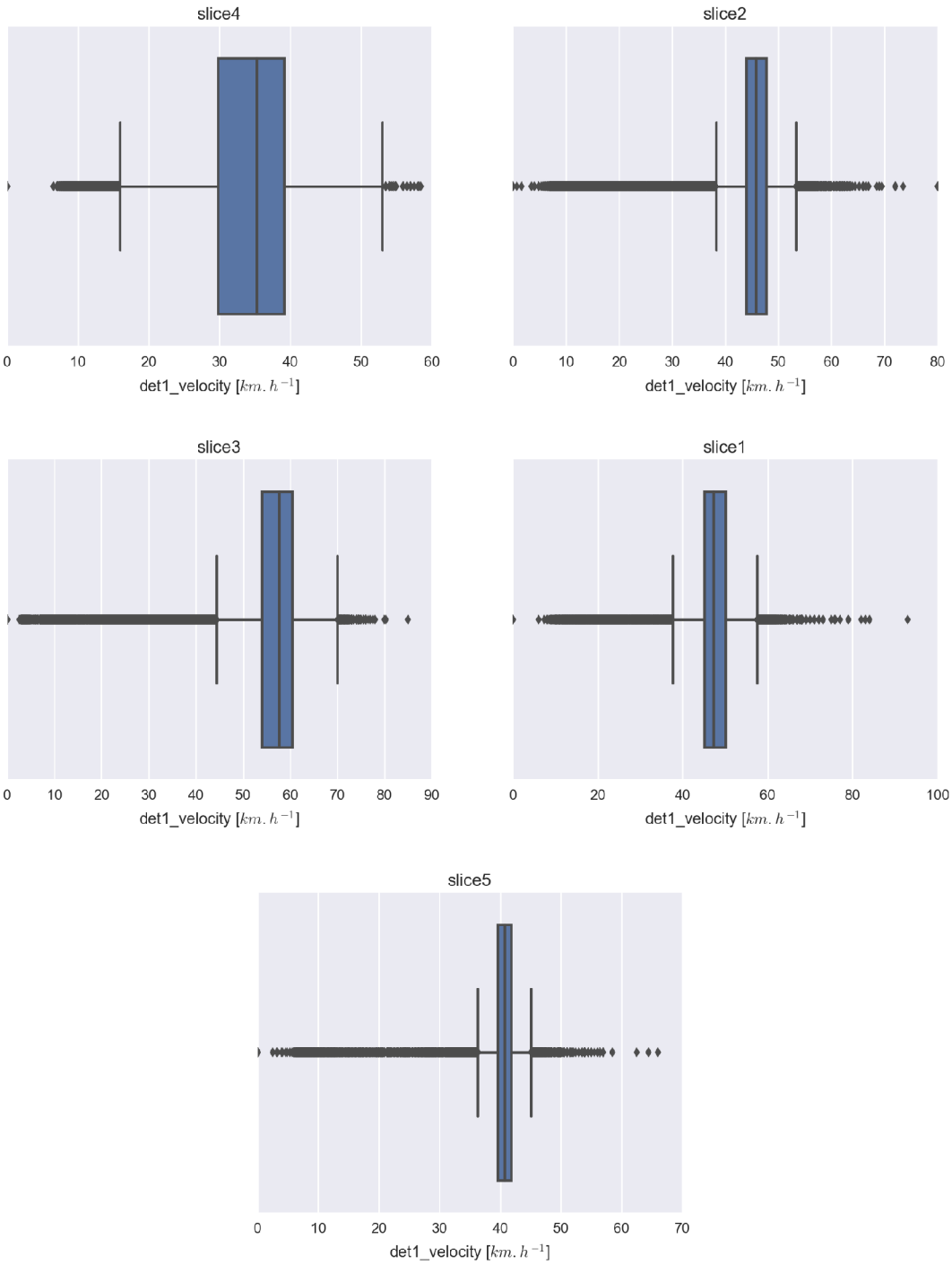
Rozloženie hodnôt premennej **det2_count** je ukázané na obrázku 5.4. Agregácia dát z jednotlivých senzorov patriacich k jednému senzorovému rezu (slice-u) prebiehala ako súčet hodnôt.

5.2 Analýza okamžitých a reálnych hodnôt dojazdových dôb

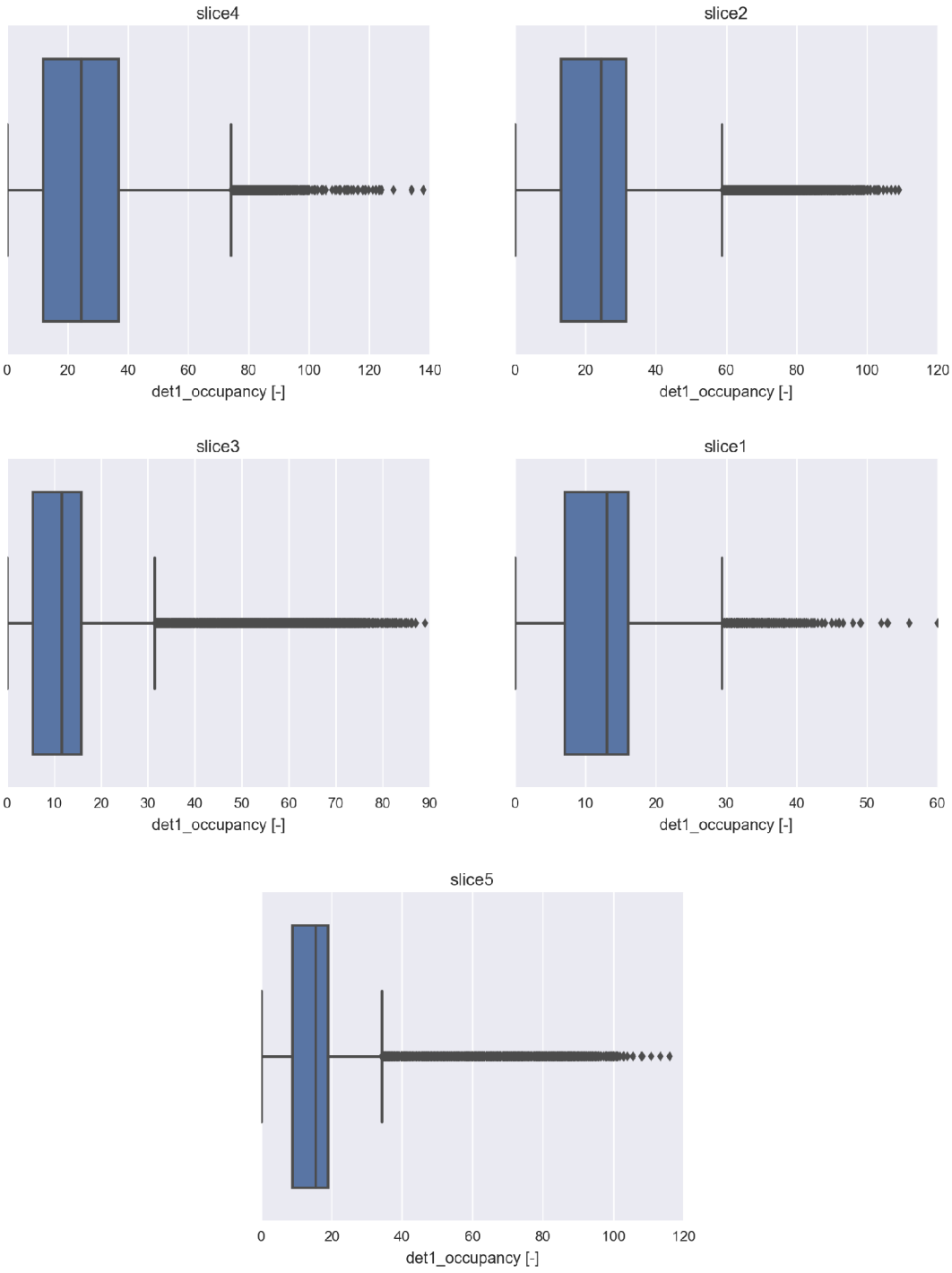
Na obrázku 5.5 je viditeľné rozloženie hodnôt spätne dopočítaných dojazdových dôb v čase pracovného týždňa a víkendu. Priebehy spätne dopočítaných dojazdových dôb pre jednotlivé dni v týždni pre viacero týždňov sú zobrazené na obrázku 5.6 (rôznymi farbami sú znázornené jednotlivé týždne).

Z rozloženia hodnôt dojazdových dôb je zrejmé, že počas víkendov je dojazdová doba pomerne stabilná (IQR≈170 s mediánom ≈620). V 14% prípadov presahuje dojazdová doba hornú hranicu pásma daného Tukeyho testom normálnosti hodnôt (3. kvartil + 1.5 × IQR) a sú teda považované za *outliers*. Je preto otázne, či majú byť tieto hodnoty z dát odstránené alebo použité (potenciálne môžu niesť cenné informácie).

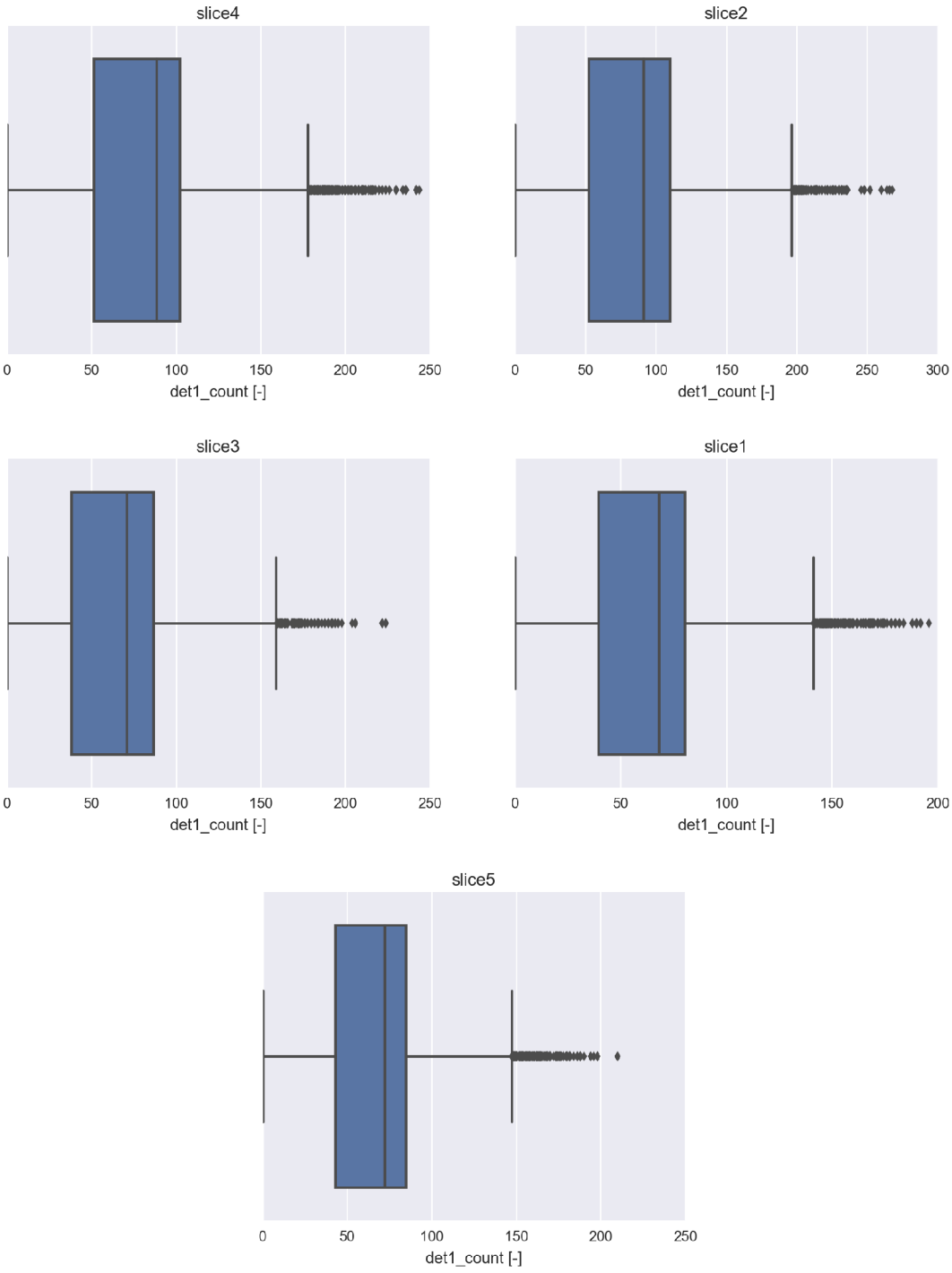
Na rozdiel od víkendov má dojazdová doba počas pracovných dní oveľa väčší rozptyl s IQR≈480 s mediánom ≈770. Horná hranica Tukeyho testu normality má hodnotu ≈1250, a rozloženie vykazuje *outliers* vo vačšom rozsahu hodnôt ako počas víkendov (19% zaznamenaných hodnôt). Vzhľadom na ich rozptyl a povahu dát zaznamenaných počas pracovných dní je však s veľmi veľkou pravdepodobnosťou nesú informácie o priebehoch dojazdových dôb počas dopravných špičiek a teda mali by byť zachované.



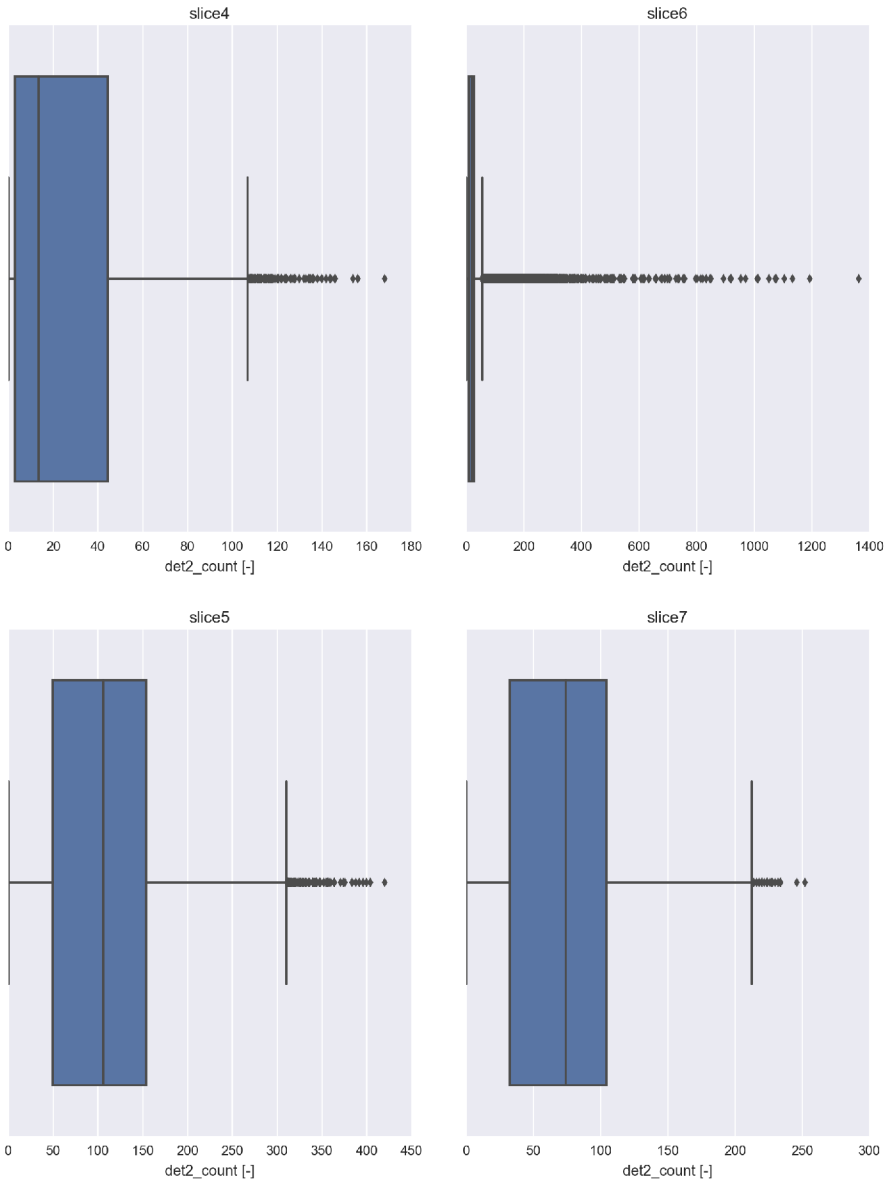
Obr. 5.1: Rozloženie premennej **det1_velocity** pre jednotlivé senzorové rezy v úseku



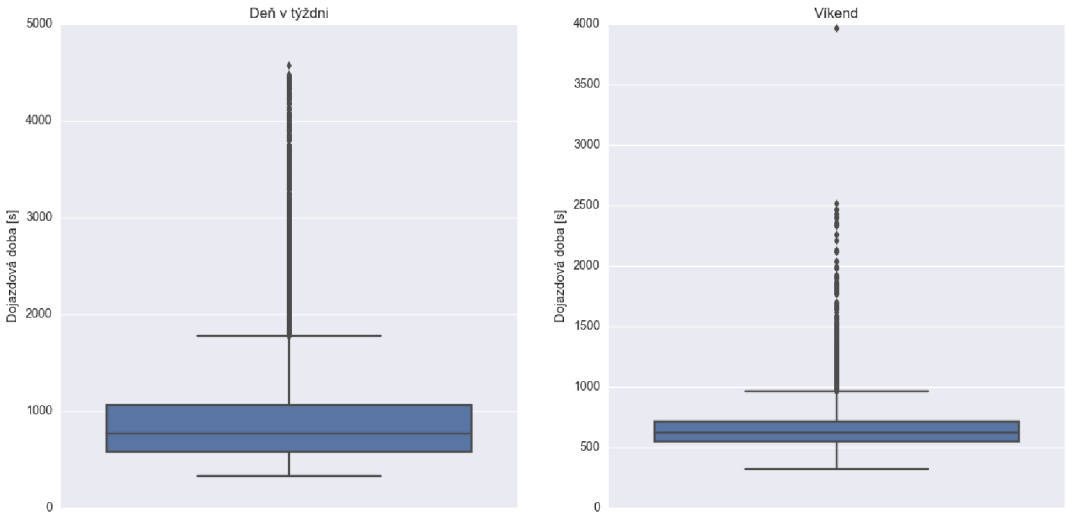
Obr. 5.2: Rozloženie premennej **det1_occupancy** pre jednotlivé senzorové rezy v úseku



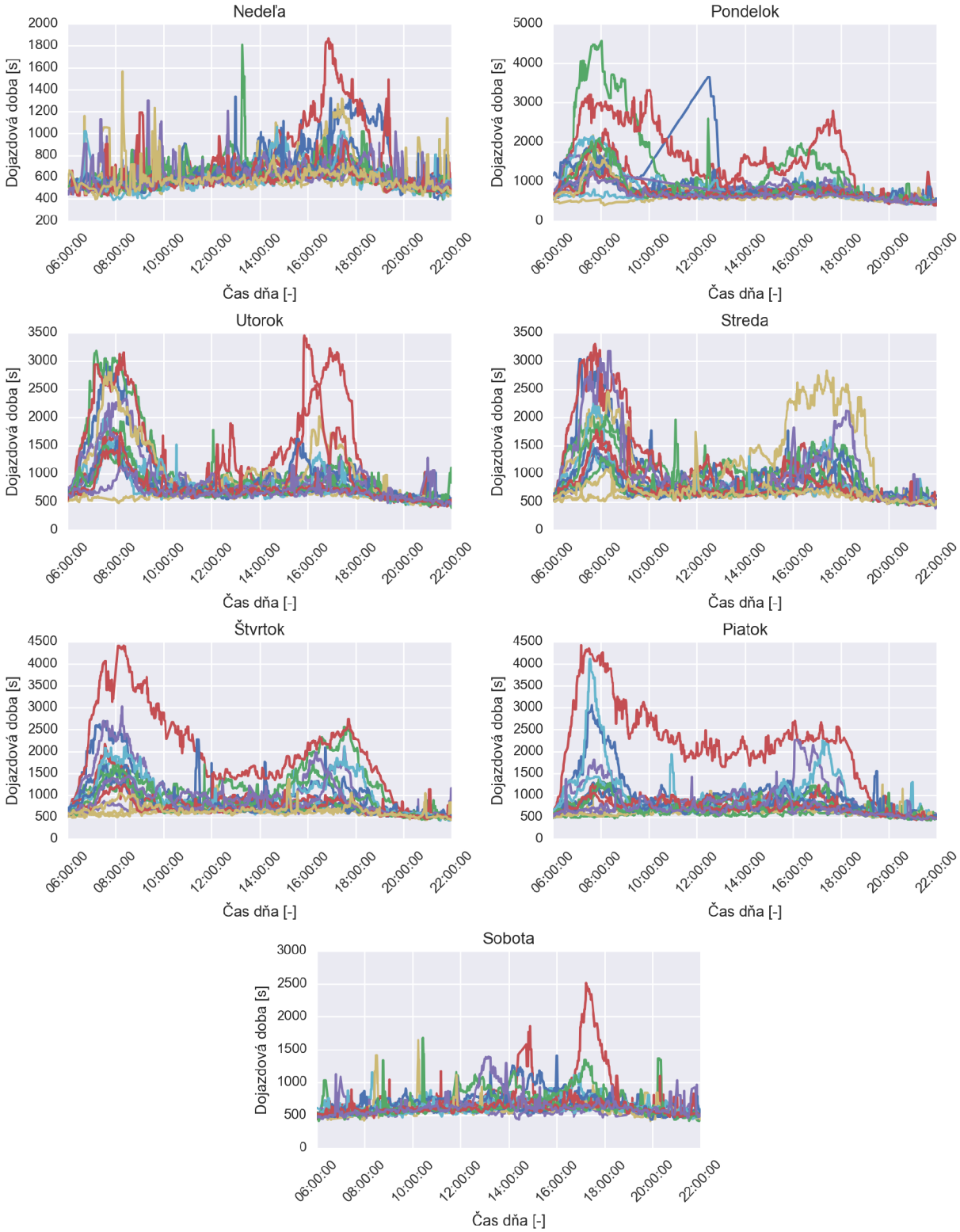
Obr. 5.3: Rozloženie premennej **det1_count** pre jednotlivé senzorové rezy v úseku



Obr. 5.4: Rozloženie premennej **det2_count** pre jednotlivé senzorové rezy v úseku



Obr. 5.5: Porovnanie rozloženia hodnôt dojazdových dôb pre víkendy a pracovné dni



Obr. 5.6: Porovnanie priebehov dojazdových dôb pre jednotlivé dni v týždni

6 NÁVRH PRÍZNAKOV A PREDIKČNÝCH MODELOV

V tejto kapitole bude popísaný návrh predikčných modelov. Jedným z najdôležitejších krokov pri tvorbe modelov je výber príznakov, preto bude popísaná aj ich tvorba a overenie ich prediktívnej hodnoty. Overenie ich vhodnosti bude prebiehať za pomoci modulu predstaveného v kapitole 7.

Po vytvorení príznakov budú predstavené možné modely, ktoré by mohli byť použité na predikciu dojazdových dôb.

6.1 Výber vhodných príznakov a ich tvorba

Základom dobrého modelu strojového učenia je výber príznakov, ktoré vhodne reprezentujú dané dáta. Sú to príznaky, ktoré sú vysoko korelované s výstupnou premennou (v tomto prípade dojazdovou dobou).

V predchádzajúcej kapitolách (viď kapitola 5) sú popísané dáta, ktoré sú k dispozícii na tvorbu modelov. Avšak tieto dáta môžu byť ďalej spracované, čím sa z nich môžu extrahovať užitočné príznaky.

V nasledujúcich podkapitolách bude popísaný výber príznakov a ich tvorba, následne bude zistená korelácia vytvorených príznakov s výstupnou premennou a tiež ich vzájomná korelácia. Okrem toho bude algoritmicky zostavený rebríček príznakov s najväčšou výpovednou hodnotou.

6.1.1 Tvorba príznakov

Ako už bolo zmienené v podkapitole 4.2, niektoré dáta môžu mať chybné hodnoty - označené "-1". Pri predspracovaní budú tieto dáta nahradené platnými hodnotami interpolovaním medzi dvomi najbližšími platnými hodnotami.

V rámci tvorby príznakov boli vytvorené príznaky popísané nižšie. Príznaky sú tvorené transformáciami vstupných dát tak, aby bol pre každý časový údaj získaný príznakový vektor z tých dát, ktoré danému časovému údaju prislúchajú. Transformácie hodnôt senzorových skupín na jednu hodnotu sú analogické s transformáciami využitými pri analýze dát v podkapitole 5.1.

det1_velocity

Tento príznak je vytvorený na základe hodnôt rýchlostí zaznamenaných skupinou senzorov na senzorovom reze - premenná **det1_velocity**. Transformácia všetkých

hodnôt rýchlostí, zaznamenaných senzorovou skupinou na reze, na jednu reprezentatívnu hodnotu - príznak - je vykonaná aplikovaním aritmetického priemeru.

det1_occupancy

Tento príznak je vytvorený na základe hodnôt obsadenosti cestnej komunikácie, ktoré sú vyprodukované senzormi na senzorickom reze - premenná **det1_occupancy**. Príznaky sú vytvorené ako súčet obsadeností pre danú skupinu senzorov.

det1_count

Tento príznak je vytvorený na základe hodnôt počtu vozidiel, ktoré boli zaznamenané skupinou senzorov na senzorickom reze komunikáciou za určitý čas. Využitá je premenná **det1_count**. Dáta zo senzorickej skupiny, ak je prítomných viac senzorov, sú agregované - ako súčet počtov vozidiel.

det2_count

Tento príznak je vytvorený na základe hodnôt počtu vozidiel, ktoré boli zaznamenané skupinou senzorov (indukčné smyčky) na senzorickom reze komunikáciou za určitý čas. Využitá je premenná **det2_count**. Dáta zo senzorickej skupiny, ak je prítomných viac senzorov, sú agregované - ako súčet počtov vozidiel.

tt_bck

Príznak **tt_bck** je založený len na okamžitej dojazdovej dobe. Z obrázku 6.1 je patrné, že táto hodnota je dostatočne blízka reálnej hodnote dojazdovej doby a preto bola vybraná ako jeden z príznakov.

Dni v týždni

Jedná sa o viacero príznakov (**Weekend, Monday, Tuesday, Wednesday, Thursday, Friday**) reprezentujúcich, ktorý deň v týždni bolo meranie uskutočnené. Príznaky sú teda založené na premennej **calculation_time_local**. Tieto príznaky sú na rozdiel od vyššie popísaných príznakov binárne.

Vytvorenie tohto typu príznakov je inšpirované obrázkom 5.6, z ktorého je patrné, že pre každý deň v týždni a víkend existuje v priebehu dojazdových dôb určitý vzor.

Obr. 6.1: Porovananie okamžitej dojazdovej doby (**bck_tt**) a reálnej dojazdovej doby (**tt_real**)

day_of_week a time_of_day

Tieto dva príznaky sú podobne ako predchádzajúci príznak založené na premennej **calculation_time_local**, avšak s tým rozdielom, že nie sú binárne, ale môžu nadobúdať niektorú z vopred definovaných hodnôt.

- Príznak **day_of_week** nadobúda hodnoty $1 \ldots 7$ pre pondelok až nedeľu, reprezentuje deň v týždni, v ktorom boli dáta zaznamenané.
- Príznak **time_of_day** nadobúda hodnoty of 0...23, reprezentuje hodinu, v ktorej boli dáta zaznamenané.

Tieto príznaky boli, podobne ako predchádzajúce príznaky vychádzajúce z časového údaju, inšpirované obrázkom 5.6.

time_bin

Tento príznak je tiež založený na časovom údaji **calculation_time_local** a poskytuje oproti predchádzajúcim príznakom jemnejšie rozdelenie času. Rozdeľuje dni na 10 minútové časové okná a teda môže nadobúdať hodnoty $0 \ldots 143$. Má hodnotu časového okna, v ktorom boli zaznamenané dáta zo senzorov.

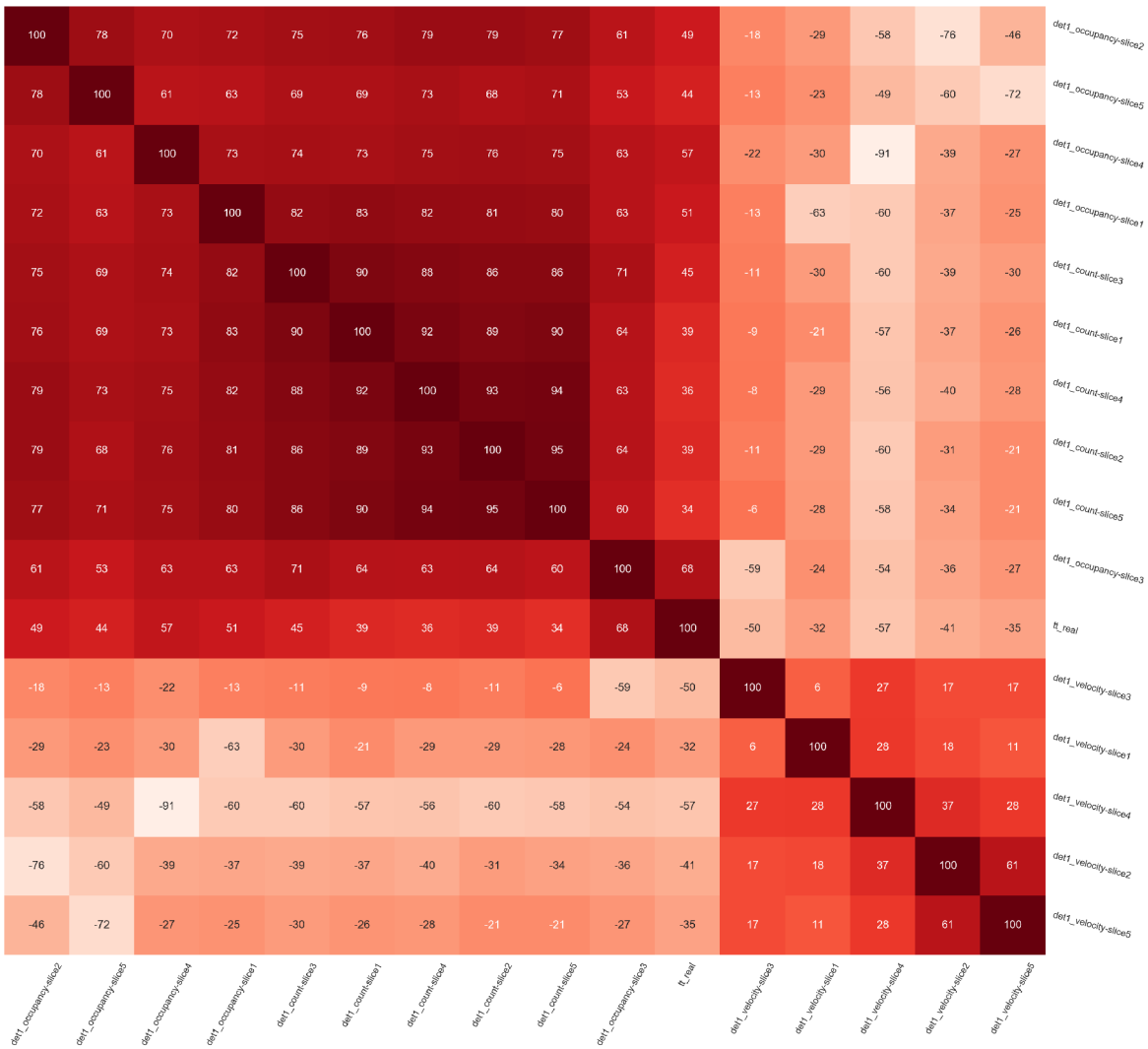
6.1.2 Overovanie prediktívnej hodnoty príznakov

V podkapitole 6.1.1 bolo vytvorených viacero príznakov. Ďalším krokom je zistenie, ktoré z týchto vytvorených príznakov majú prediktívnu hodnotu, teda ktoré príznaky by mali byť zapojené do modelu na predikciu dojazdových dôb. Testovanie bude

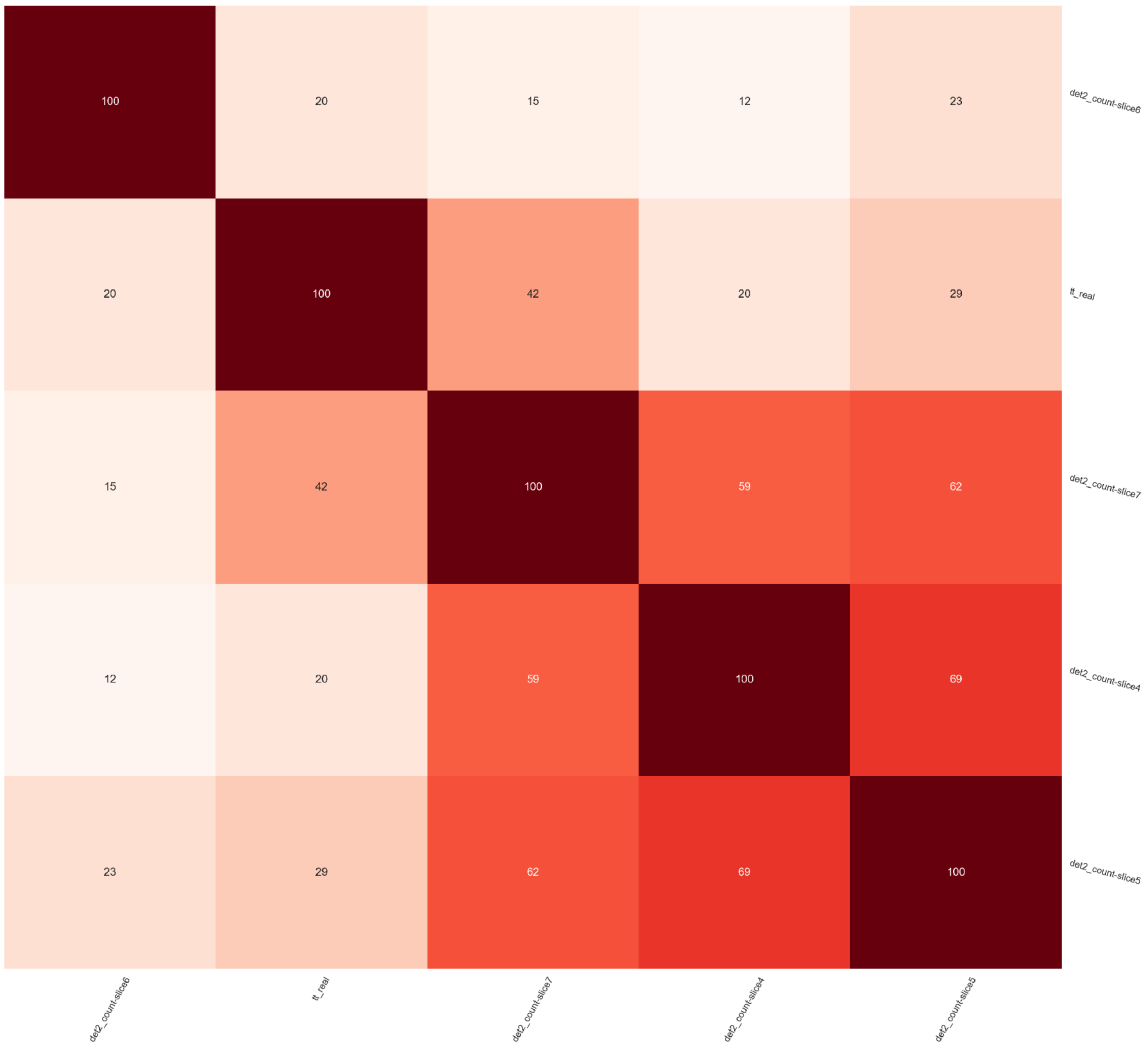
prebiehať prostredníctvom vytvoreného testovacieho modulu, ktorého architektúra je popísaná v kapitole 7.

Pre znázornenie, ktoré príznaky by mali byť zapojené do modelu môžeme použiť korelačnú maticu - s využitím funkcie `plot_correlation` (podkapitola 7.2.2). Na nasledujúcich obrázkoch (6.2, 6.3, 6.4) sú zobrazené korelačné matice príznakov predstavených v predchádzajúcej kapitole spolu s výstupnou premennou - hodnotou dojazdovej doby - **tt_real**.

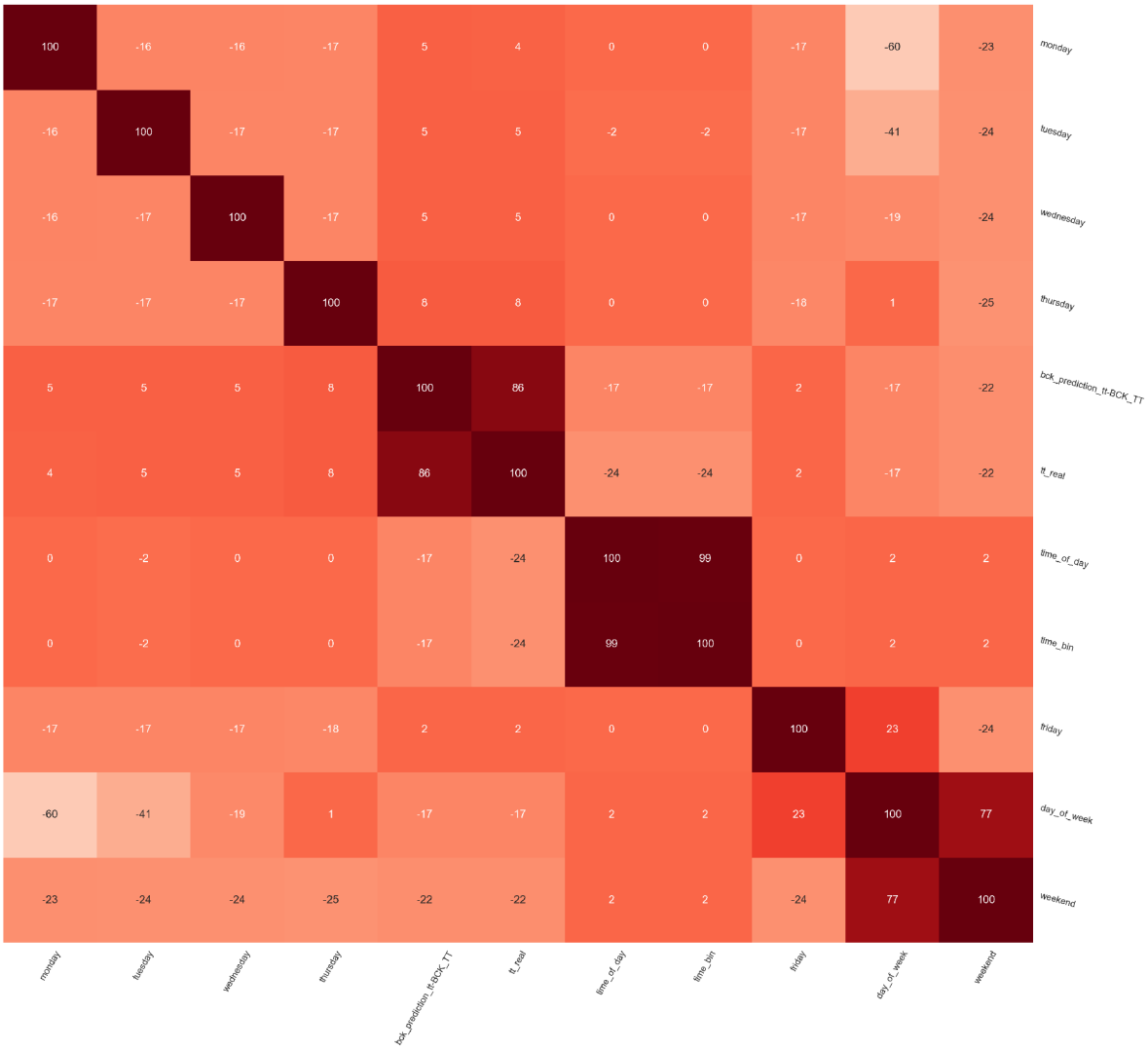
Je potrebné podotknúť, že fakt, že príznak je nekorelovaný s výstupnou veličinou - skutočnou dojazdovou dobou, nemusí nutne znamenať, že neexistuje žiadna závislosť medzi hodnotou príznaku a hodnotou výstupnej veličiny, keďže korelácia vyšetruje iba lineárny vzťah.



Obr. 6.2: Korelačná matica príznakov vyplývajúcich zo senzorov **det1** a výslednej dojazdovej doby



Obr. 6.3: Korelačná matica príznakov vyplývajúcich zo senzorov **det2** a výslednej dojazdovej doby



Obr. 6.4: Korelačná matica príznakov vyplývajúcich časových údajov a výslednej dojazdovej doby

Zo zobrazenia korelačných matíc je zrejmé, ktoré príznaky najviac korelujú so skutočnou dojazdovou dobou **tt_real**. Tieto sú uvedené v tabuľke 6.1.

Je možné, že medzi vybranými príznakmi z tabuľky 6.1 sú príznaky, ktoré sú navzájom korelované. Pri vysoko korelovaných príznakoch je žiadúce ponechať iba jeden z nich, keďže ponechanie ostatných by viedlo na model s vyššou komplexitou, pričom je možné, že presnosť tohto modelu by bola obdobná ako presnosť modelu pri odstránení korelovaných príznakov.

Avšak, pri modeli, ktorý tvorí funkcia viacerých premenných môže nastať situácia, že premenná, ktorá je nekorelovaná s výstupnou veličinou bude mať vysokú váhu pri použití v modeli. Preto nie je možné korelačnú maticu použiť ako jediný ukazovateľ dôležitosti príznakov.

Ďalšou možnosťou ako vyhodnotiť vhodné príznaky je použitie funkcie implementovanej v rámci testovacieho frameworku - `feature_evaluation` (viď podkapitola 7.2.1). Výstupom tejto funkcie je zoznam príznakov z danej skupiny, ktorý je zobrazený v tabuľke 6.2.

Príznak	Absolútna hodnota korelácie
bck_prediction_tt-BCK_TT	86
det1_occupancy-slice3	68
det1_occupancy-slice4	57
det1_velocity-slice4	57
det1_occupancy-slice1	51
det1_velocity-slice3	50
det1_occupancy-slice2	49
det1_count-slice3	45
det1_occupancy-slice5	44
det2_count-slice7	42
det1_velocity-slice2	41
det1_count-slice2	39
det1_count-slice1	39
det1_count-slice4	36
det1_velocity-slice5	35
det1_count-slice5	34
det1_velocity-slice1	32
det2_count-slice5	29
time_bin	24
time_of_day	24
weekend	22
det2_count-slice4	20
det2_count-slice6	20
day_of_week	17
thursday	8
wednesday	5
tuesday	5
monday	4
friday	2

Tab. 6.1: Výber príznakov korelovaných s **tt_real**

Príznak	Významnosť príznaku
bck_prediction_tt	0.61
det1_occupancy-slice3	0.52
monday	0.50
weekend	0.47
friday	0.38
tuesday	0.36
det1_velocity-slice4	0.35
det1_velocity-slice3	0.34
time_bin	0.31
wednesday	0.31
thursday	0.31
det1_occupancy-slice1	0.30
day_of_week	0.29
time_of_day	0.26
det1_velocity-slice1	0.24
det1_count-slice3	0.23
det2_count-slice7	0.21
det1_velocity-slice2	0.21
det1_count-slice4	0.19
det1_occupancy-slice2	0.13
det1_occupancy-slice4	0.11
det1_count-slice5	0.09
det2_count-slice4	0.08
det1_count-slice2	0.06
det1_velocity-slice5	0.05
det1_occupancy-slice5	0.03
det1_count-slice1	0.02
det2_count-slice5	0.01
det2_count-slice6	0.00

Tab. 6.2: Signifikantné príznaky vybrané funkciou `feature_selection`

6.1.3 Výber príznakov

Na základe výsledkov z prechádzajúcej kapitoly môže byť môžu byť vybrané len niektoré, tie najdôležitejšie príznaky s najvyššou prediktívnou hodnotou (ak je takýto krok potrebný s ohľadom na použitý regresný algoritmus).

Je potrebné podotknúť, že tieto príznaky sú signifikantné pre jednu definovanú sekciu a iné sekcie môžu mať odlišné signifikantné príznaky. Pre optimálne nastavenie je teda potrebné nájsť pre každú sekciu jej optimálne príznaky.

6.2 Predikčné modely

V predchádzajúcej podkapitole boli popísané príznaky, ktoré môžu byť použité pri tvorbe modelov. Táto podkapitola sa už zaoberá samotnými modelmi, ktoré môžu byť použité na predikciu dojazdových dôb. Pri návrhu jednotlivých modelov boli do úvahy vzaté výsledky vedeckých prác, ktoré sa zaoberali problematikou predikcie dojazdových dôb, ktoré sú popísané v kapitole 3.

Navrhnutými modelmi sú:

- **TimeDomainModel**
- **ClusterModel**
- **CombinedModel**

Podstata týchto modelov bude v nasledujúcom texte rozobraná.

6.2.1 TimeDomainModel

TimeDomainModel je model založený na časovom pôvode dát, ktorý je inšpirovaný algoritmom LOKRR popísanom v podkapitole 3.2.

Model využíva predpoklad, že dojazdová doba je závislá od času dňa - tento čas je reprezentovaný príznakom **time_bin** (podkapitola 6.1.1).

Dáta - príznaky zo všetkých dní z celého obdobia použitého na tréning algoritmu sú teda rozdelené do N časových okien $w_i \quad i = 0 \ldots N-1$ reprezentujúcich 10 minútový časový úsek. Táto fáza je vykonaná na základe hodnôt **time_bin** (ich hodnota je $0 \ldots N$).

Pre každé časové okno w_i je vytvorený prediktor, ktorý zaisťuje predikciu iba pre toto špecifické okno. Pre tréning tohto prediktoru sú použité okrem dát z časového okna w_i aj dáta z priľahlých okien - w_{i-1} a w_{i+1}. Týmto je vytvorené tolerančné pásmo pre nepravidelnosti v doprave na komunikáciách, kedy v niektoré dni môže očakávaný vzor nastať s istým posunom a už by nespadal do očakávaného časového okna a teda aj oblasti pôsobnosti prediktoru.

Ako jednotlivé prediktory pre časové okná je možné použiť ľubovoľný regresný algoritmus.

Hlavným rizikom (a zároveň výhodou) tohto modelu je to, že dáta sú rozdelené podľa času a jednotlivé prediktory sú trénované na úzkom výbere dát špecifických pre daný čas dňa. Toto síce umožňuje lepšie zachytenie ich štruktúry, ale zároveň znemožňuje adekvátnu reakciu na nový podnet, ktorý sa v množine použitej na tréning prediktoru nevyskytoval. Preto môže dochádzať k veľkým odchýlkam od reality, ak sa vzory v doprave pravidelne nevyskytujú v rovnakých časoch.

6.2.2 ClusterModel

Tento model je založený na tom, že existujú vektory príznakov (prislúchajúce dojazdovým dobám), ktoré sú si natoľko podobné, že tvoria clustre - zhluky. Pri predpoklade, že takéto zhluky dátových bodov existujú, je výhodnejšie vytvoriť prediktor pre každý cluster.

Takto je docielené, že podobne ako pri modeli **TimeDomainModel**, je predikcia v závislosti na vstupných dátach, vykonaná prediktorom špecificky prispôsobeným na tieto dáta - v tomto prípade je rozhodovacím faktorom príslušnosť ku clusteru.

V prípade tohto modelu je potrebné vybrať clusterizačný algoritmus, ktorým budú vytvorené clustre (pre fázu tréningu modelu) a ktorý dokáže priradiť príslušnosť nových dát ku clusterom (fáza predikcie). Tieto algoritmy sú popísané v podkapitole 2.3.

Tiež je potrebné stanoviť regresný algoritmus, ktorý bude využitý na vytvorenie prediktorov pre jednotlivé clustre.

ClusterModel je model asi s najmenšími rizikami vyplývajúcimi z princípu návrhu, dát a nepravidelných vzorov v doprave. Vzhľadom na to, že dáta nie sú diferencované podľa pevného kľúča, ale podľa ich vzájomnej blízkosti v nadpriestore, sú body reprezentujúce podobné javy zaradené do toho istého clusteru. Týmto odpadá problém popísaný u modelu **TimeDomainModel**. Na vytvorenie modelu je ale použitých viacero prediktorov, čím je pravdepodobné, že model bude lepšie prispôsobený vstupným dátam.

6.2.3 CombinedModel

Tento model využíva kombináciu N plnohodnotných modelov (modelov dojazdových dôb popísaných v predchádzajúcich podkapitolách) a spracováva hodnoty dojazdovej doby predikované týmito modelmi. Na kombináciu výsledkov jednotlivých modelov

môže byť použitá ľubovoľná agregačná funkcia, v najjednoduchšom prípade priemer týchto hodnôt, pri ktorom je výsledná predikcia $\hat{y}$ stanovená ako

$$\hat{y} = \frac{1}{N} \sum_{i=0}^{N} f_i(X) \tag{6.1}$$

Jedno z rizík tohto modelu predstavuje problém trénovania modelu a síce hľadania vhodných hyperparametrov pre jednotlivé modely. Týka sa to hlavne časovej a výpočtovej náročnosti, čo je zrejmé z tabuliek uvedených v podkapitole 8.2.

7 VÝVOJ TESTOVACIEHO FRAMEWORKU

Pre analýzu dát, príznakov a posúdenie presnosti modelov a jej vyhodnotenie bol vyvinutý testovací framework. Tento zapúzdruje všetky potrebné manipulačné a vyhodnocovacie funkcie, ktoré boli pri analýze využité. Testovací framework predstavuje modul **test_suite** vytvorený v programovacom jazyku Python.

7.1 Výber platformy

Pri výbere platformy - programovacieho jazyka - pre testy a predikčný systém bolo prihliadnuté na viacero kritérií. Medzi ne patrí výkon, efektivita programovania, dostupnosť knižníc, zrozumiteľnosť, užívateľská základňa a multiplatformnosť.

Spomedzi kandidátov Python, R, C# a C++ bol vybraný Python z dôvodov popísaných nižšie. Medzi ďalšie dôvody patrí tiež fakt, že autor práce má skúsenosti s vývojom systému využívajúceho strojové učenie v jazyku Python.

Python Programovací jazyk Python bol zvolený z toho dôvodu, že v súčasnej doba sa jedná o jeden z najpoužívanejších jazykov pre dátovú analýzu a implementáciu systémov s využitím strojového učenia. Z toho vyplýva množstvo knižníc a modulov vytvorených pre túto tematiku, ktoré sú voľne dostupné a využiteľné. Taktiež je to jazyk vhodný na rýchle prototypovanie, ale aj tvorbu komplexných riešení. Zároveň sa jedná o open source riešenie.

Väčšina knižníc pre strojové učenie je úplne alebo čiastočne implementovaná v jazykoch Fortran a C++, čo prináša vysokú výpočetný výkon, a zároveň ľahko použiteľná využívajúc rozhrania implementované v Pythone. Jedná sa o knižnice (moduly) `sklearn` [25], `numpy` [26], `pandas` [27]. Pre Python je tiež k dispozícii prepracovaná vizualizačná knižnica `matplotlib` [28].

R R je, podobne ako Python, jazyk s veľkou zásobou knižníc na dátovú analýzu a strojové učenie. Je tiež open source a má výkonné knižnice. Avšak, na rozdiel od Pythonu nie je príliš používané pri produkčných nasadeniach aplikácií a vytvorenie komplexného programu produkčnej verzie je náročnejšie.

C++ Výkon C++ je v porovnaní s Pythonom násobne vyšší, pokiaľ Python nepoužíva knižnice vytvorené v C++. Avšak vývoj systému v C++ ako jazyka nižšej úrovne než Python, vyžaduje násobne viac času a je omnoho náročnejší. Ďalším problémovým krokom je vizualizácia výsledkov.

C# V prípade tohto programovacieho jazyka je efektivita a rýchlosť vývoja vyššia než u C++. Nevýhodou je menšia užívateľská základňa pre aplikácie typu dátová analýza a strojové učenie a taktiež menší výber knižníc než u jazyka Python.

V tejto kapitole bude modul **test_suite** popísaný, spolu so zameraním využitých knižníc (modulov) jazyka Python.

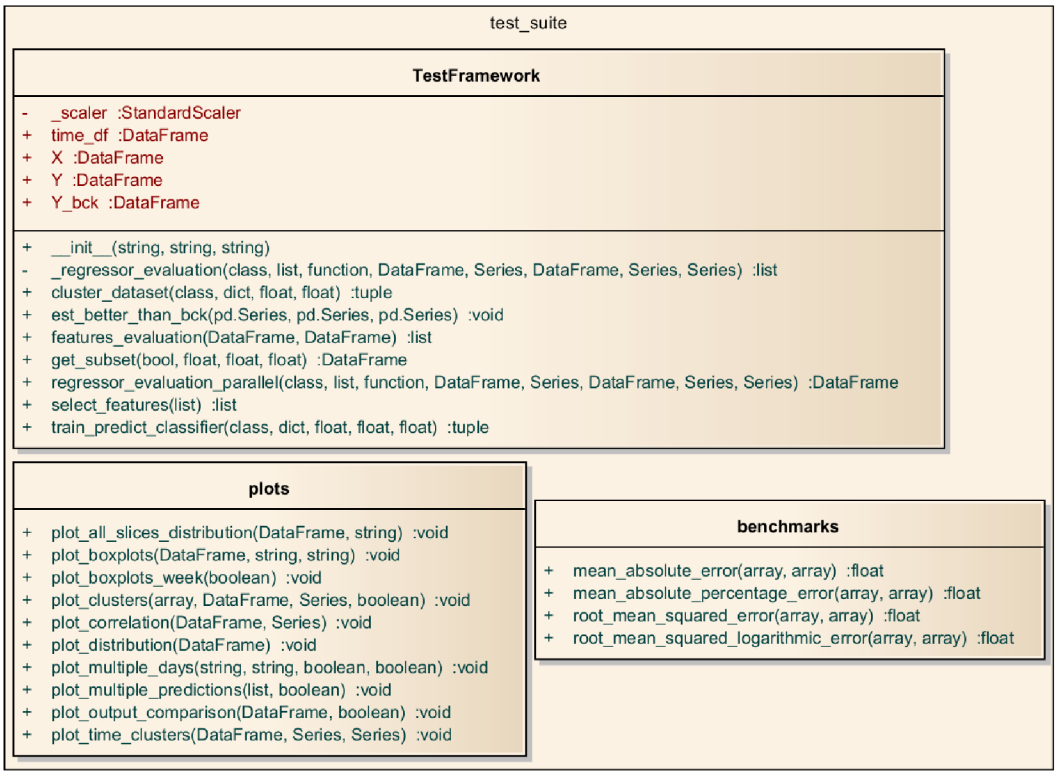
7.2 Architektúra

Modul **test_suite** sa skladá z troch hlavných častí (obrázok 7.1):

TestFramework - trieda, ktorá zapuzdruje metódy potrebné na manipuláciu s dátami, tvorbu modelov a ich vyhodnocovanie

plots - súhrn funkcií pre vykresľovanie a vizualizáciu dát

benchmarks - súhrn benchmarkov - funkcií, ktoré môžu byť použité ako ukazovatele výkonnosti modelov



Obr. 7.1: Architektúra modulu **test_suite**

7.2.1 TestFramework

TestFramework v sebe zapuzdruje všetky dôležité metódy pre manipuláciu s dátami, vyhodnocovanie významu príznakov, testovanie modelov a iné. Obsahuje tieto

metódy:

- **`_regressor_evaluation`** - vyhodnocovanie modelov
- **`cluster_dataset`** - vytvorenie clustrov z poskytnutého datasetu
- **`est_better_than_bck`** - vyhodnotenie kvality predikcie metódy v porovnaní s okamžitou dojazdovou dobou za určitý časový úsek
- **`features_evaluation`** - ohodnotenie dôležitosti príznakov, ktoré sú v datasete. Využíva viacero algoritmov - Random Forests, Linear Regression, Lasso, Ridge Regression a iné. Tieto algoritmy sú použité k vytvoreniu modelu daného datasetu a následne sú z týchto algoritmov zistené hodnoty dôležitosti príznakov v daných modeloch (algoritmus musí toto ohodnocovanie podporovať). Výsledná hodnota dôležitosti premennej je priemerom týchto hodnôt.
- **`get_subset`** - získanie dát na tréning, test a validáciu
- **`regressor_evaluation_parallel`** - metóda paralelizujúca **`_regressor_evaluation`**
- **`select_features`** - vybranie určených príznakov z datasetu
- **`train_predict_classifier`** - vytvorenie modelu na jednej časti dát a predikovanie na zvyšku, ktorý nebol použitý pri tréningu - *cross validation*

7.2.2 plots

Táto časť testovacieho frameworku slúži na vizualizáciu dát, je založená na knižnici `matplotlib` [28] a jej nadstavbe. Obsahuje nasledujúce funkcie:

- **`plot_all_slices_distribution`** - vykreslenie rozloženia hodnôt príznaku zo všetkých rezov (slice-ov) úseku
- **`plot_boxplots`** - vykreslenie rozloženia príznakov prostredníctvom boxplot-ov
- **`plot_boxplots_week`** - vykreslenie rozloženia pomocou boxplot-ov pre pracovné dni a víkendy zvlášť
- **`plot_clusters`** - vizualizácia dojazdových dôb ako bodov, ktoré prislúchajú k určitým clustrom, ktoré vytvárajú vektory príznakov - dátové body.
- **`plot_correlation`** - vykreslenie korelačnej matice príznakov
- **`plot_distribution`** - funkcia na vykreslenie rozloženia hodnôt príznakov
- **`plot_multiple_days`** - zobrazenie referenčných dojazdových dôb pre všetky dni v týždni
- **`plot_multiple_predictions`** - zobrazenie priebehov predikcií od viacerých modelov do jedného grafu tak, aby ich bolo možné porovnať
- **`plot_output_comparison`** - vizualizácia porovnania rozdielov medzi skutočnou dojazdovou dobou **tt_real** (podkapitola 4.1.4) a okamžitou dojazdovou dobou **bck_prediction_tt** (podkapitola 4.1.3)
- **`plot_time_clusters`** - vykreslenie hodnôt dojazdových dôb ako bodov, ktoré

prislúchajú ku clusterom vytvoreným na základe rozdelenia dát do časových okien

7.2.3 benchmarks

Táto časť testovacieho frameworku slúži vyhodnocovanie výsledkov predikcií modelov. Je založená na podkapitole 2.4; jednotlivé implementované funkcie majú za vzor metriky popísané v spomenutej podkapitole. Implementované sú tieto funkcie (metriky):

- `mean_absolute_percentage_error`
- `root_mean_squared_logarithmic_error`
- `mean_absolute_error`
- `root_mean_squared_error`

8 POROVNANIE MODELOV

Táto kapitola sa zaoberá porovnaním modelov podľa viacerých kritérií - jedná sa o kritéria kvality predikcie popísané v podkapitole 2.4 a tiež výkonnostné kritériá (využitie systémových prostriedkov) - napr. časy potrebné na natrénovanie (vytvorenie modelov) a predikciu.

Pri tomto vyhodnotení boli využité dáta pochádzajúce z viacerých úsekov, aby sa overila celková úspešnosť a prínos predikcie vzhľadom na súčasne používaný model.

Porovnávaniu modelov predchádzala implementácia týchto modelov v jazyku Python s využitím knižníc `pandas` [27], `sklearn` [25] a `numpy` [26]. Pri behu testov bol využitý modul **test_suite** (kapitola 7).

8.1 Nájdenie parametrov modelov

Vzhľadom na to, že modely dojazdových dôb popísané v kapitole 6 sú parametrické, je dôležité ich správne nastaviť, t.j. nájsť parametre, ktoré s ktorými majú najlepší výkon. Hľadanie hyperparametrov prebiehalo individuálne pre každý úsek s využitím funkcie `regressor_evaluation_parallel`.

Pre použitie v modeloch boli testované algoritmy Ridge, ElasticNet a lineárna regresia, nakoľko v priebežných testoch vykazovali pomerne vysokú presnosť predikcie. Algoritmy SVR, k-NN v týchto testoch vykazovali pomerne zlé výsledky - horšie ako súčasne používaný model. Tréning regresoru založeného na SVR navyše vyžaduje pomerne veľký výpočetný výkon a teda aj čas potrebný na naučenie, čo je nežiadúce.

Pre clustering (ak ho model implementoval) bol použitý algoritmus K-Means. Algoritmus DBSCAN nebol použitý z dôvodu extrémnych nárokov na operačnú pamäť.

8.2 Testy modelov na úsekoch

Na nasledujúcich grafoch a v tabuľkách je sú znázornené predikcie jednotlivých modelov v porovnaní so súčasne využívaným modelom predikcie (podkapitola 4.1.3) a zároveň reálnou dojazdovou dobou (podkapitola 4.1.4).

Avšak, testy v tejto kapitole sú vykonané na predspracovaných dátach, a výsledky testov môžu byť odlišné od výsledkov v reálnych podmienkach pri behu predikčného systému. Testy predikcie pri behu predikčného systému sú diskutované v podkapitole 9.6.

8.2.1 Úsek 1

Tento úsek s dĺžkou 6.58 km je najdlhší z testovaných úsekov. Výsledky testu sú uvedené v tabuľkách 8.1 a 8.2.

Pre tento úsek sa ukázal ako najvhodnejší model **ClusterModel** s nastavením pre vytvorenie 5-tich clusterov. Regresory pre jednotlivé clustre boli vytvorené na báze algoritmu Ridge s parametrom $\lambda = 10$. Podľa metriky MAPE má model o 19 % vyššiu presnosť ako momentálne používané riešenie postavené na okamžitej dojazdovej dobe.

Na obrázku 8.1 je zobrazené porovnanie predikcií modelov počas rannej špičky.

Model	MAPE [%]	RMSLE [-]	MAE [s]	RMSE [-]	R^2 [-]
BCK	10.88	0.16	77.61	147.37	0.48
TimeDomainModel	10.52	0.15	75.24	127.36	0.61
ClusterModel	8.81	0.13	64.63	114.17	0.69
CombinedModel	9.17	0.14	66.16	118.94	0.66

Tab. 8.1: Porovnanie modelov aplikovaných na úsek 1 vzhľadom k reálnej dojazdovej dobe

Model	Čas tréningu [s]	Čas predikcie [s]
BCK	NaN	NaN
TimeDomainModel	0.98	2.65
ClusterModel	0.49	3.03
CombinedModel	1.71	5.62

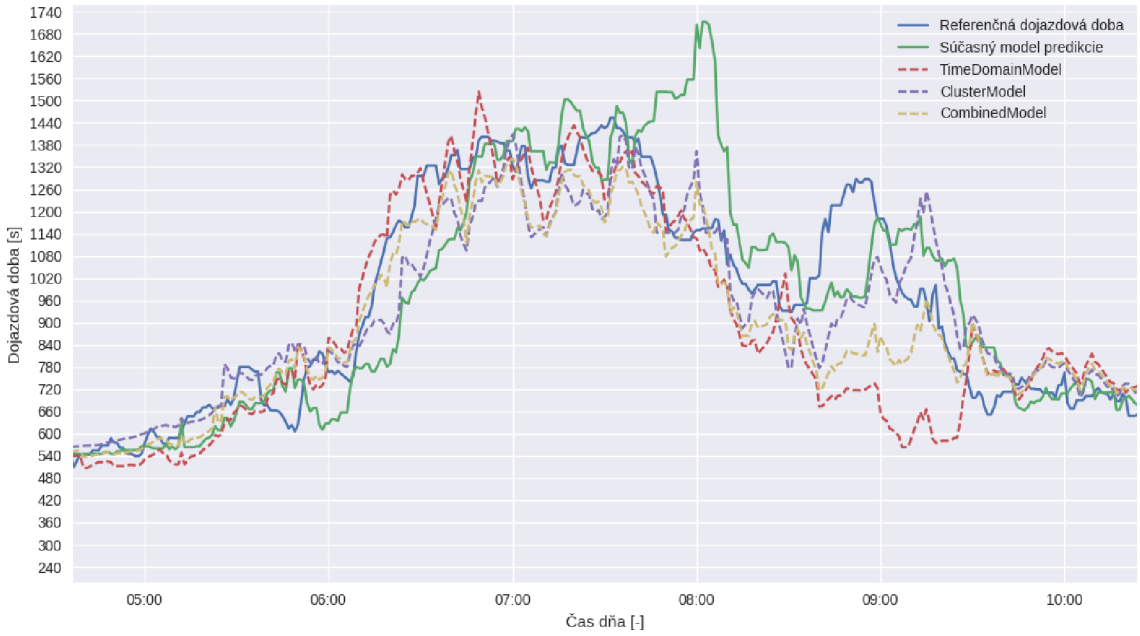
Tab. 8.2: Porovnanie modelov aplikovaných na úsek 1 vzhľadom k času potrebného na tréning a predikciu

8.2.2 Úsek 2

Jedná sa o úsek s dĺžkou 1.53 km, z toho dôvodu sú dojazdové časy relatívne krátke. Výsledky testu sú uvedené v tabuľkách 8.3 a 8.4. Na obrázku 8.2 je uvedený príklad predikcie model na tomto úseku počas rannej špičky.

Podobne ako v prípade prvého úseku, aj tu dosiahol najlepšie výsledky **ClusterModel**, ktorý podľa metriky MAPE dosiahol o 80 % lepšie výsledky ako v súčasnosti používané riešenie.

Kombinácia parametrov najlepšieho riešenia bola 50 clusterov; regresory clusterov boli postavené na algoritme Ridge s parametrom $\lambda = 50$.



Obr. 8.1: Porovnanie predikcií modelov počas rannej špičky na úseku 1

Model	MAPE [%]	RMSLE [-]	MAE [s]	RMSE [-]	R^2 [-]
BCK	35.14	0.32	37.57	45.89	0.82
TimeDomainModel	10.17	0.14	14.10	26.41	0.94
ClusterModel	7.02	0.11	11.31	26.14	0.94
CombinedModel	8.08	0.12	11.93	24.23	0.95

Tab. 8.3: Porovnanie modelov aplikovaných na úsek 2 vzhľadom k reálnej dojazdovej dobe

8.2.3 Úsek 3

Posledný testovací úsek je stredne dlhý úsek s dĺžkou 4.2 km. Výsledky testu sú uvedené v tabuľkách 8.5 a 8.6. Priebeh predikcie počas rannej špičky vybraný z dát pre test je ukázaný na obrázku 8.3.

V tomto prípade sa tiež najlepšie osvedčil model **ClusterModel** s algoritmom Ridge, tesne nasledovný modelom **CombinedModel**. Kombinácia parametrov, ktorá dosiahla najvyššiu presnosť, bola 25 clusterov a $\lambda = 1$. Táto kombinácia mala o 15 % nižšiu chybu MAPE ako súčasne používaný model.

Model	Čas tréningu [s]	Čas predikcie [s]
BCK	NaN	NaN
TimeDomainModel	1.51	3.42
ClusterModel	0.76	3.04
CombinedModel	1.52	6.16

Tab. 8.4: Porovnanie modelov aplikovaných na úsek 2 vzhľadom k času potrebného na tréning a predikciu

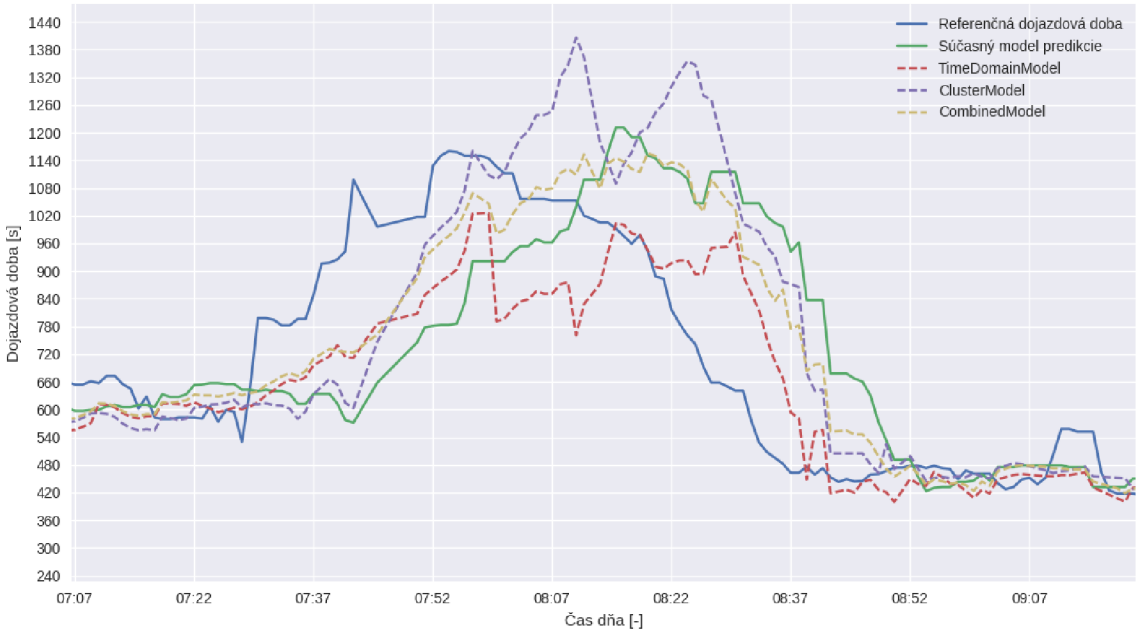
Obr. 8.2: Porovnanie predikcií modelov počas rannej špičky na úseku 2

Model	MAPE [%]	RMSLE [-]	MAE [s]	RMSE [-]	R^2 [-]
BCK	9.04	0.12	41.14	69.19	0.63
TimeDomainModel	8.73	0.12	40.01	66.65	0.66
ClusterModel	7.65	0.11	35.44	63.54	0.69
CombinedModel	7.75	0.11	35.52	61.40	0.71

Tab. 8.5: Porovnanie modelov aplikovaných na úsek 3 vzhľadom k reálnej dojazdovej dobe

Model	Čas tréningu [s]	Čas predikcie [s]
BCK	NaN	NaN
TimeDomainModel	0.69	2.44
ClusterModel	0.71	2.70
CombinedModel	1.08	5.36

Tab. 8.6: Porovnanie modelov aplikovaných na úsek 3 vzhľadom k času potrebného na tréning a predikciu



Obr. 8.3: Porovnanie predikcií modelov počas rannej špičky na úseku 3

8.3 Zhodnotenie modelov

Vzhľadom k výsledkom testov je zrejmé, že všetky aplikované modely na predikciu dojazdovej doby prekonali súčasne používaný model postavený na okamžitej dojazdovej dobe.

Vo všetkých prípadoch sa ako najlepší model (model s najnižšou chybou predikcie vzhľadom k reálnej dojazdovej dobe) preukázal model **ClusterModel** v kombinácii s algoritmom Ridge. Jeho chyba predikcie podľa metriky MAPE bola v porovnaní so súčasným modelom predikcie o 15 - 80 % nižšia. Pri zvažovaní časový metrík sa tento model taktiež ukazuje ako najlepší, pričom tréning modelu trvá menej ako 1 sekundu.

V poradí druhý najlepší model je **CombinedModel**, ktorý dosahuje o 14 - 79 % menšie chyby MAPE v porovnaní so súčasným modelom. Avšak čas potrebný na vytvorenie tohto modelu je o 52 - 249 % dlhší ako v prípade modelu **ClusterModel**.

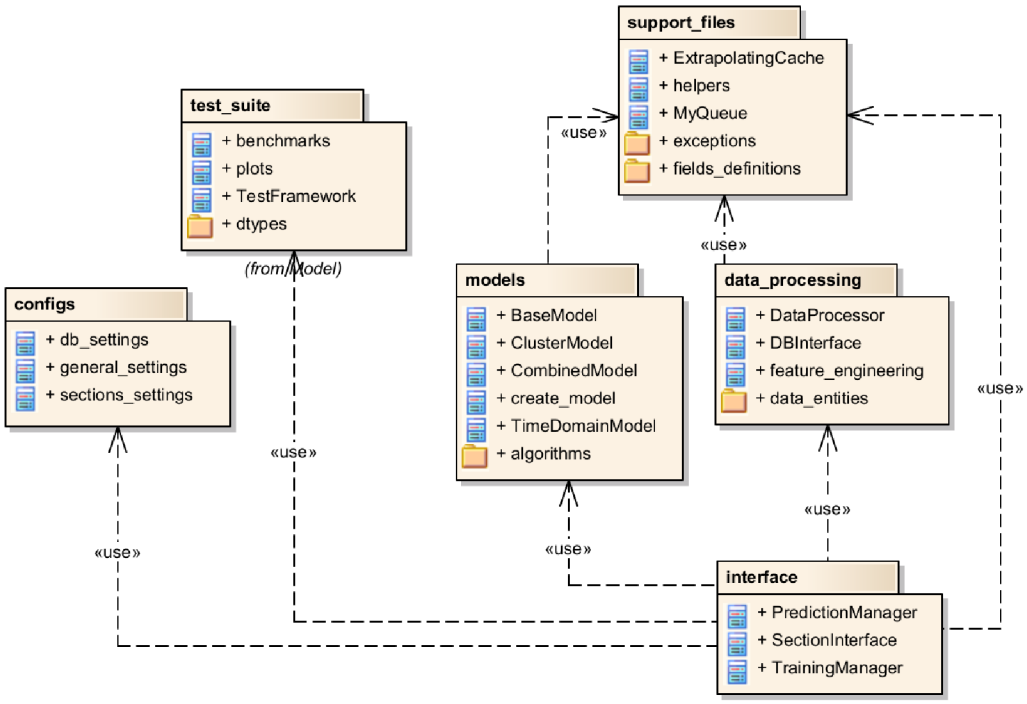
Ako najmenej úspešný model sa ukázal **TimeDomainModel** s chybou MAPE nižšou v rozmedzí 3 - 71 %. Jeho trénovacie a testovacie časy boli v priemere mierne väčšie ako u modelu **ClusterModel**.

9 PREDIKČNÝ SYSTÉM

V rámci diplomovej práce bol tiež implementovaný komplexný predikčný systém. Pri jeho návrhu bol dôraz kladený na rozšíriteľnosť - v budúcnosti sa predpokladá pridávanie ďalších typov senzorov a teda generovanie nových typov príznakov. V tejto kapitole bude priblížená jeho štruktúra a funkcionalita s využitím UML diagramov. Detaily fungovania je možné nájsť v dokumentácii ku kódu, ktorá je v prílohe diplomovej práce.

Predikčný systém je tvorený niekoľkými modulmi, ktoré zabezpečujú rôzne časti predikčného procesu. Štruktúra systému je zobrazená na obrázku 9.1, jedná sa o moduly:

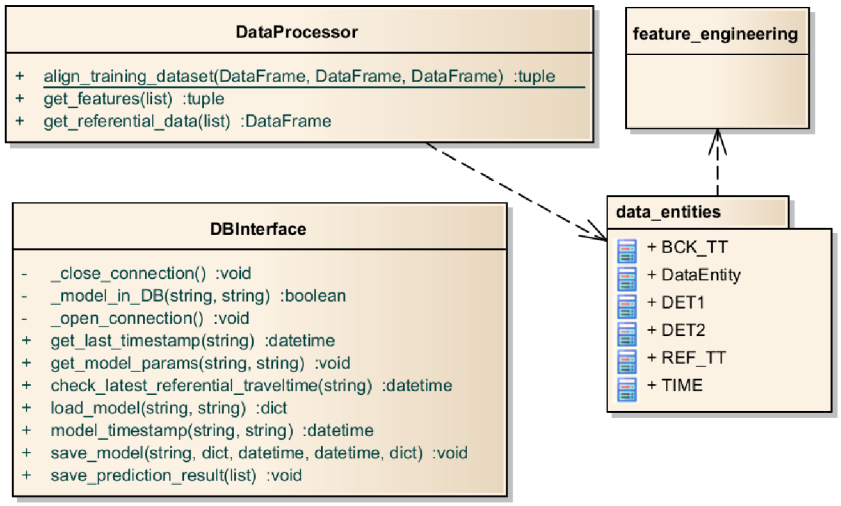
- **data_processing** - modul zodpovedajúci za zisk dát, ich spracovanie a tvorbu príznakov
- **interface** - modul obsahujúci rozhrania na prácu s predikčným systémom - predikciu a tvorbu modelov
- **models** - modul obsahujúci implementácie jednotlivých modelov dojazdových dôb, ako boli popísané v podkapitole 6.2
- **support_files** - modul s pomocnými funkciami, triedami a definíciami konštánt
- **configs** - súbory, ktoré obsahujú nastavenia predikčného systému



Obr. 9.1: Štruktúra predikčného systému

9.1 Modul data_procesing

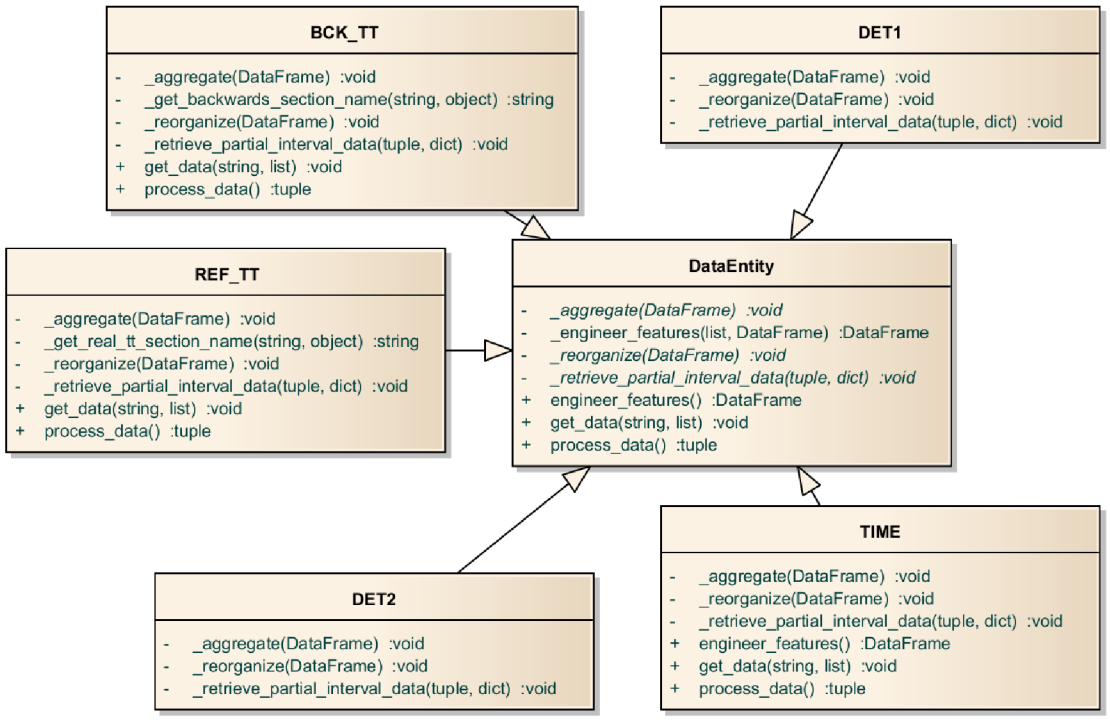
Zabezpečuje získavanie dát, ich spracovanie a tvorbu príznakov, aby dáta mohli byť následne použité pri tvorbe modelov, prípadne pri predikovaní.



Obr. 9.2: Architektúra modulu **data_processing**

Ako je zrejmé z obrázku 9.2, modul sa skladá z nasledujúcich blokov:

- **DataProcessor** - trieda zabezpečujúca získanie dát zo všetkých zdrojov, ktoré sú pre daný úsek dostupné - využíva **data_entities**
- **DBInterface** - trieda prostredníctvom ktorej je implementovaný prístup do databázy pre uloženie/načítanie modelov prediktorov, uloženie výsledkov a iné
- **data_entities** - súbor autonómnych tried pre získanie dát z databáze, ich spracovanie a generovanie príznakov. Obsahujú jednu triedu pre každú dátovú entitu - napr. snímače det1, bck_tt (okamžitá dojazdová doba), časové príznaky a iné. Prostredníctvom tejto štruktúry je možné ľahko pridať nové zdroje dát a príznaky. Štruktúra je znázornené na obrázku 9.3.
- **feature_engineering** - konfiguračný súbor s definíciami príznakov, t.j. ako a z akých dát majú byť príznaky vytvorené



Obr. 9.3: Architektúra tried na získavanie príznakov - **data_entities**

9.2 Modul interface

Pozostáva z dvoch hlavných častí:

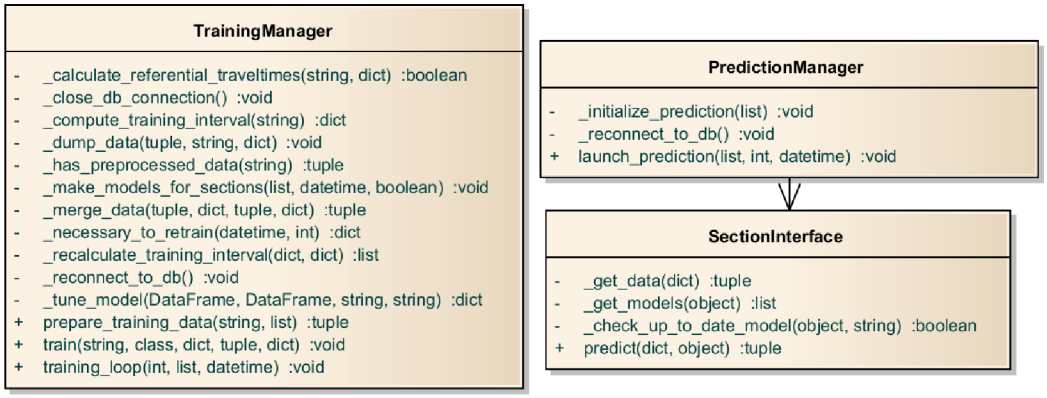
- **TrainingManager** - zabezpečuje tvorbu modelov - ich tréning
- **PredictionManager** - zabezpečuje predikciu s využitím modelov vytvorených triedou **TrainingManager**

Jeho ďalšou časťou je **SectionInterface**, ktorý poskytuje rozhranie k predikcii dojazdovej doby pre jeden úsek.

Štruktúra tohto modulu je priblížená na obrázku 9.4. Ako už bolo napísané vyššie, tento modul predstavuje rozhranie na prácu s predikčným systémom, resp. rozhranie na jeho zapojenie do aplikácie.

Všetky jeho časti - **TrainingManager**, **PredictionManager** a **SectionInterface** sú spracované tak, aby boli v súlade s požiadavkami firmy na ošetrenie krajných podmienok a tiež výkonnostné hľadiská.

V nasledujúcich podkapitolách budú tieto časti predstavené a bude popísaný princíp ich činnosti pomocou diagramov.

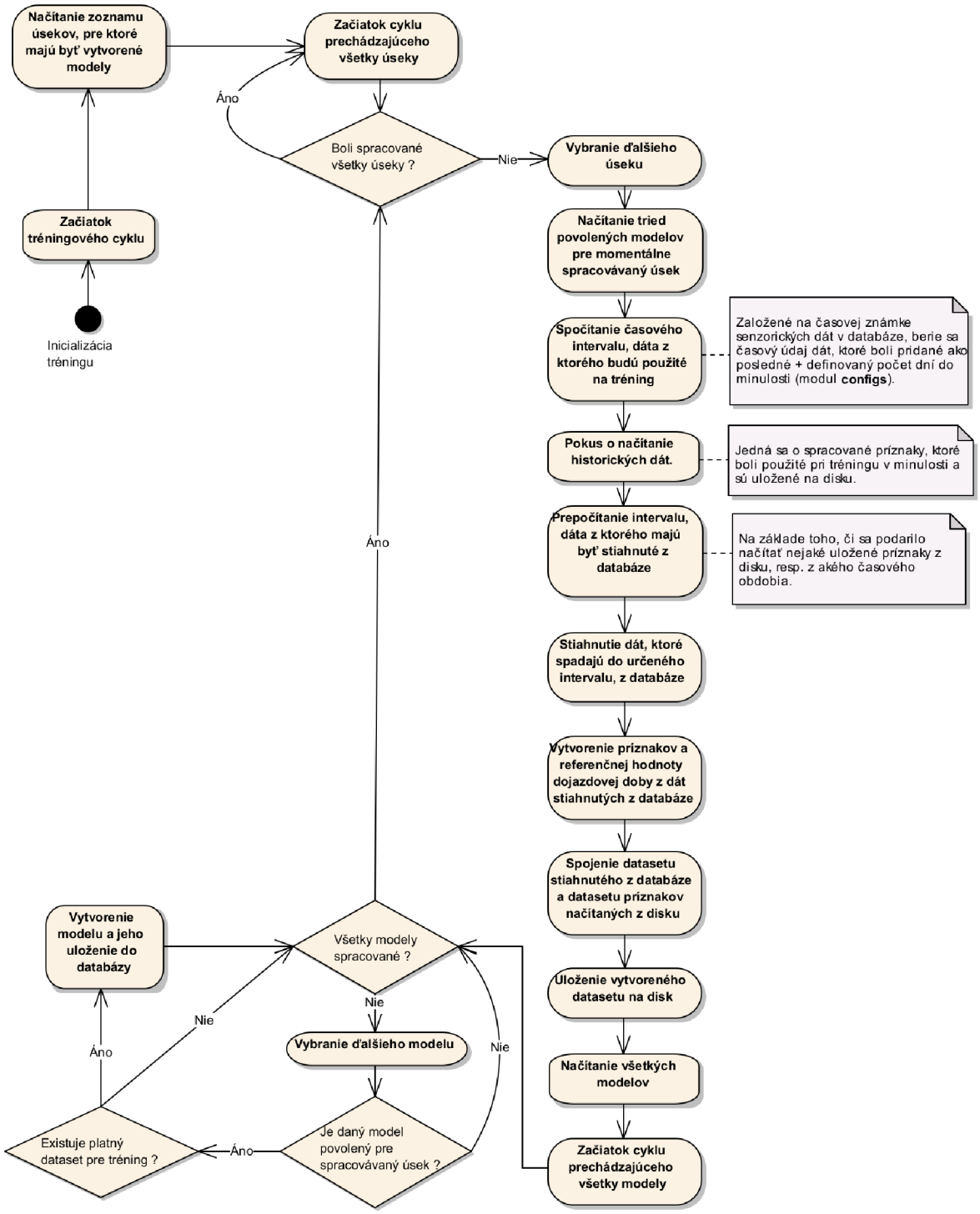


Obr. 9.4: Architektúra modulu **interface**

9.2.1 TrainingManager

Slúži na vytvorenie modelov, ktoré budú následne použité pri predikcii. Modely sú založené na historických dátach jednotlivých úsekov, resp. príznakov generovaných z týchto dát (podkapitola 6.1.1).

Tréning modelov je periodická činnosť, ktorá je kontrolovaná nastaveniami v module **configs** (podkapitola 9.5). Hlavná činnosť je reprezentovaná metódou `training_loop`, jej funkcionalita je bližšie priblížená na obrázku 9.5. Pri vytváraní modelu môže byť funkčnosť rozšírená o hľadanie hyperparametrov - čo je však veľmi výpočetne náročné.

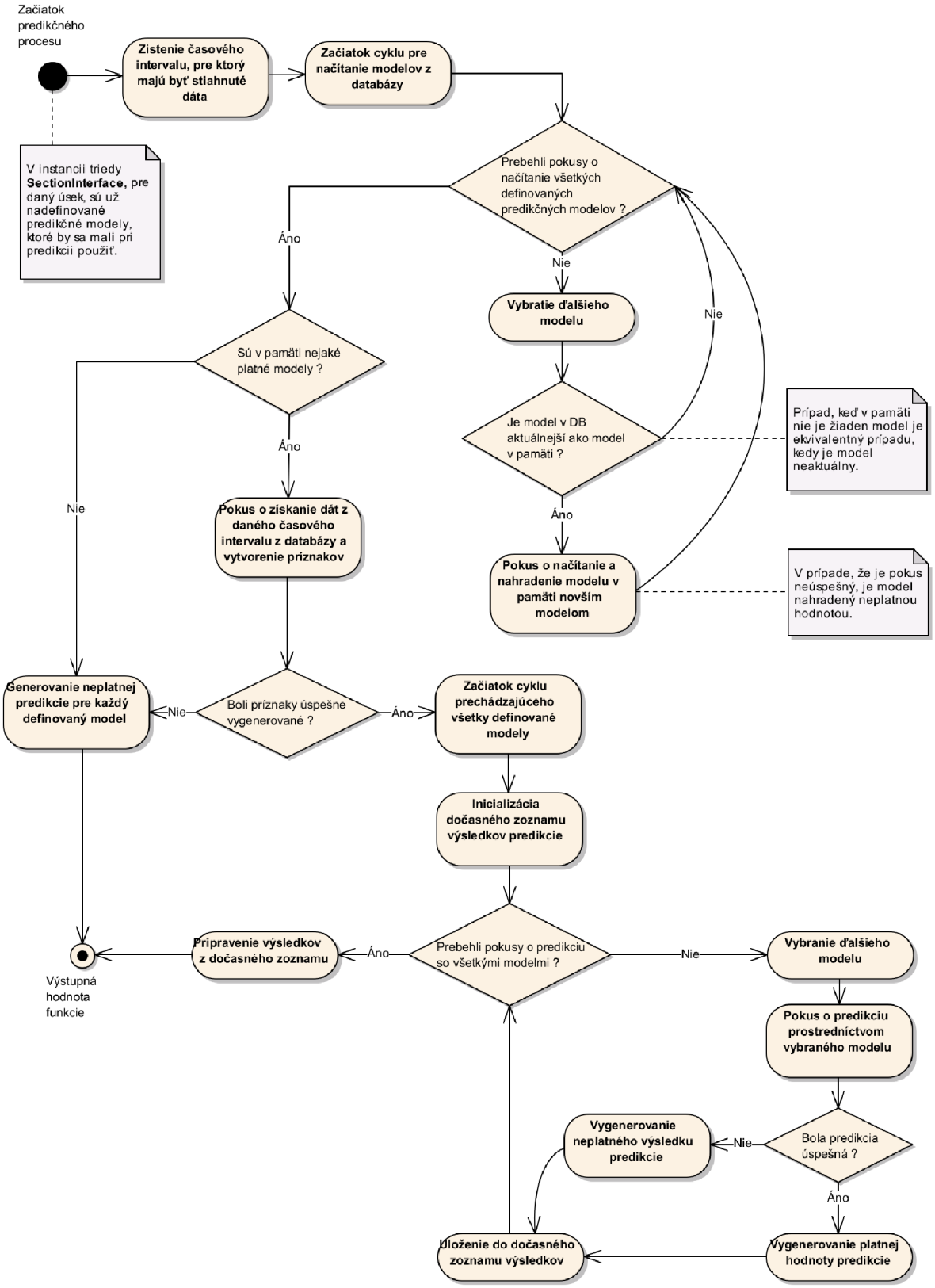


Obr. 9.5: Popis tvorby modelov implementovanej v triede **TrainingManager**

9.2.2 SectionInterface

Trieda, ktorá je využitá pri predikcii ako interface pre predikciu dojazdových dôb na jednotlivých úsekoch. Pri predikcii má každý úsek svoju vlastnú instanciu tejto triedy.

Samotný predikčný mechanizmus je implementovaný metódou `predict`, ktorej funkcionalita je znázornená na obrázku 9.6.



Obr. 9.6: Funkcionalita predikcie implementovaná triedou **SectionInterface**

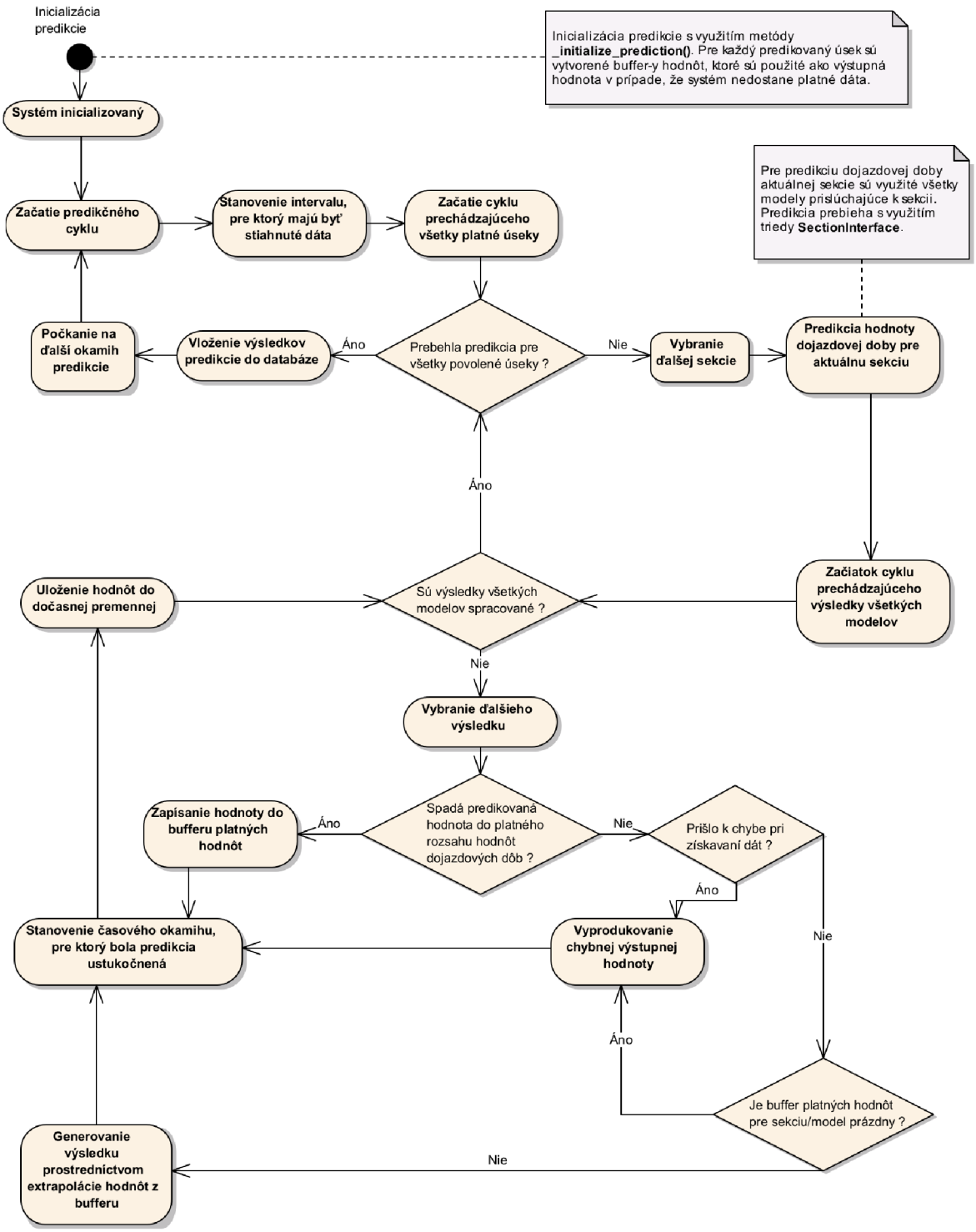
9.2.3 PredictionManager

Trieda **PredictionManager** slúži na periodickú predikciu dojazdových dôb s využitím dát, ktoré sú popísané v podkapitole 6.1.1. Diagram popisujúci jej činnosť je zobrazený na obrázku 9.7.

Na predikciu pre jednotlivé úseky využíva triedu **SectionInterface** (podkapitola 9.2.2). Pri predikcii sú tiež generované rôzne správy, ktoré vysvetľujú, čo sa pri nej stalo. Zoznam týchto správ a ich významy je možné nájsť v tabuľke 9.1.

Kód správy	Význam
MODEL_NOT_IN_DB	V databáze nebol nájdený žiadny model, ktorý by mohol byť použitý.
DATA_NOT_IN_DB	Z databázy neboli získané žiadne platné dáta.
PREDICTION_UNSUCCESSFUL	Predikcia neúspešná, viď ďalšie správy pre bližšie objasnenie príčiny.
DATA_AGGREGATION_FAILED	Agregácia dát bola neúspešná.
FEATURE_ENGINEERING_FAILED	Tvorba príznakov bola neúspešná.
RESULT_EXTRAPOLATED	Kvôli neúspešnej predikcii bola ako výsledná hodnota použitá hodnota extrapolovaná na základe historický hodnôt predikcie.
TOO_HIGH_TRAVELTIME	Hodnota dojazdovej doby bola zmenená na zmysluplnnú, pretože pôvodná hodnota vychádzajúca z predikcie bola extrémne vysoká, čo je nepravdepodobné.
PREDICTION_SUCCESSFUL	Predikcia bola vykonaná úspešne.

Tab. 9.1: Správy generované pri predikcii



Obr. 9.7: Popis princípu predikcie implementovanej v triede **PredictionManager**

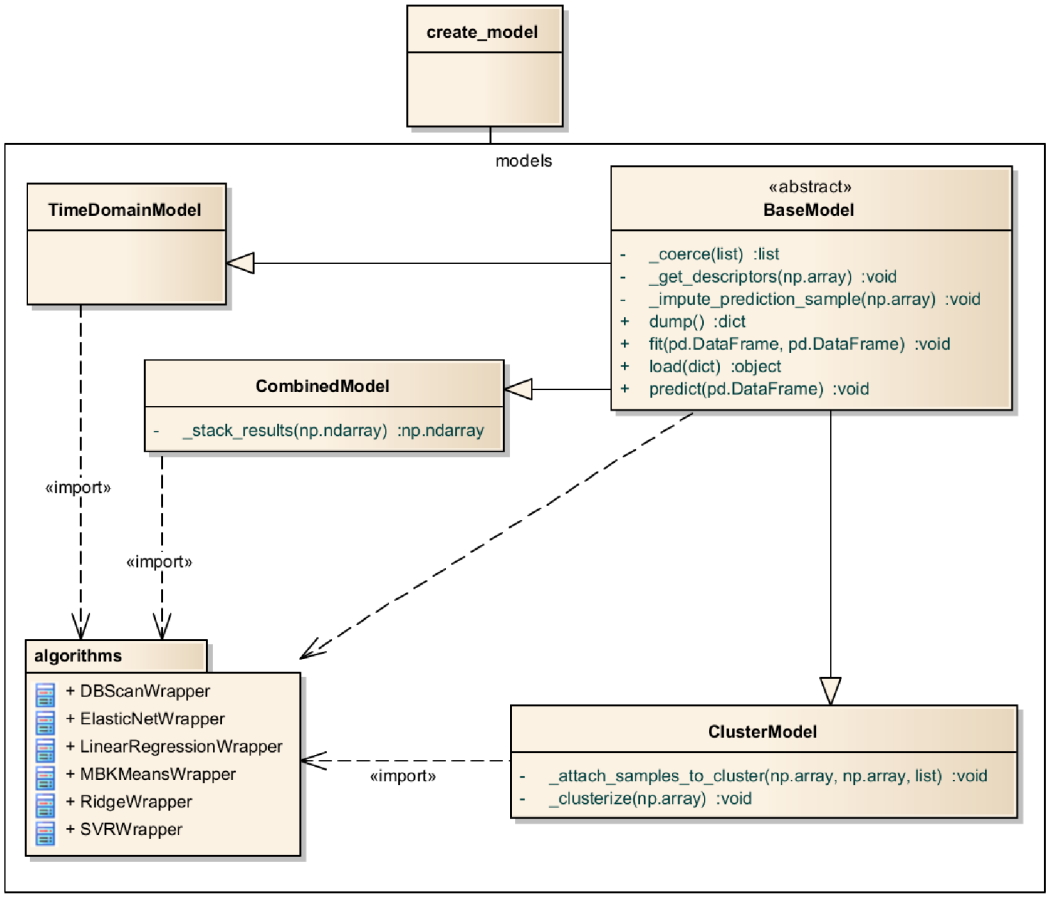
9.3 Modul models

Modul **models** zoskupuje triedy, ktoré implementujú modely popísané v podkapitole 6.2. V tomto module sú zhrnuté aj rôzne algoritmy, ktoré sú v týchto modeloch použité. Modely a algoritmy sú odvodené od triedy `BaseModel`, ktorá definuje rozhranie pre ukladanie a načítavanie modelov. Pre načítavanie uložených modelov je použitá funkcia `create_model`.

Samotné modely sú ukladané v textovej podobe - ako objekt JSON, ktorý obsahuje všetky potrebné údaje k rekonštrukcii modelu tak, aby bol použiteľný pre predikciu.

Tieto reťazce sú pri ukladaní do databázy komprimované s využitím knižnice `zlib`. Výsledná veľkosť reťazcov uložených do databázy sa líši podľa modelov a pri testovaných prípadoch dosahuje 250 kB.

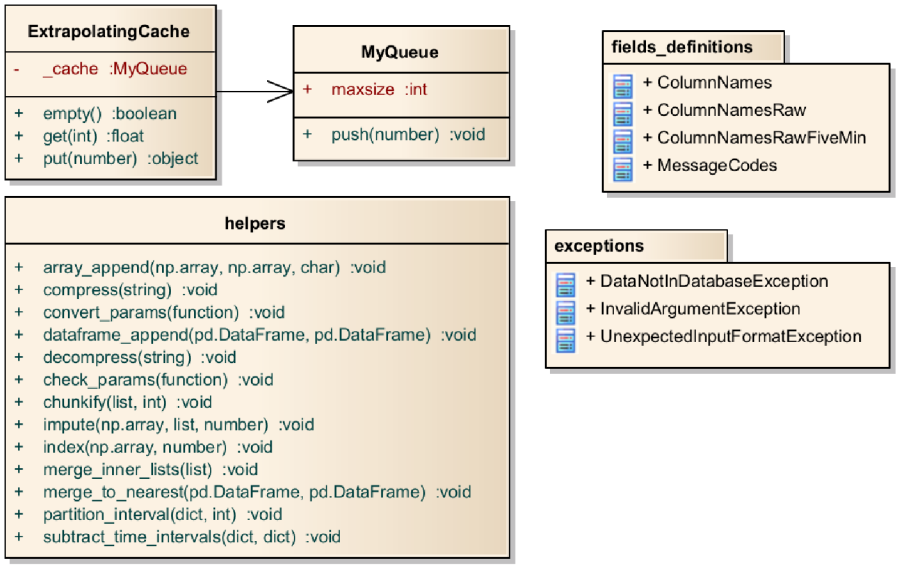
Štruktúra modulu je uvedená na obrázku 9.8.



Obr. 9.8: Architektúra modulu **models**

9.4 Modul support_files

V tomto module sú zoskupené viaceré pomocné funkcie, triedy a definície konštánt, ktoré boli potom využité pri implementácii ostatných modulov. Štruktúra tohto modulu je znázornená na obrázku 9.9. Popisy jednotlivých funkcií je možné nájsť v dokumentácii, ktorá je v prílohe.



Obr. 9.9: Štruktúra modulu **support_files**

9.5 Modul configs

Zoskupuje konfiguračné nastavenia systému. Skladá sa z troch častí:

- **db_settings** nastavenie pripojenia k databázam - databáza pre ukladanie modelov vytvorených pre jednotlivé sekcie, výsledkov, načítavanie dát a iné. Týmito nastaveniami sa de facto konfiguruje pripojenie do existujúcej infraštruktúry firmy CAMEA - k databázam, v ktorých sú uložené dáta použité pri tréningu a predikcii a zároveň k databáze, do ktorej sa ukladá výstup predikčného systému - predikované hodnoty dojazdovej doby
- **general_settings** Nastavenia týkajúce sa predikčného systému ako celku - intervaly predikcie, tréningu, cesty k uloženým dátam, modely použité predikčným systémom ...
- **section_settings** nastavenia týkajúce sa jednotlivých sekcií - senzory, ktoré sú použité na úsekoch, modely, ktoré môžu byť využité pri tvorbe modelov sekcií, časové pásma, v ktorých sa úseky a mnohé iné ...

Detailné popisy sú uvedené v priloženej dokumentácii.

9.6 Test predikčného systému

Testy uvedené v kapitole 8 sa venujú vyšetreniu presnosti a použiteľnosti modelov. Okrem týchto testov však musí byť vykonaný aj test funkčnosti predikčného systému ako celku vrátane vyhodnotenia predikcie pri produkčnom nasadení.

Tento test bol vykonaný na tých istých úsekoch komunikácií ako testy modelov v kapitole 8. Pri tomto produkčnom teste bol systém nastavený tak, aby pre tréning modelov boli použité dáta z posledných 7/15/30/60 dní, model bol pretrénovaný každých 24 hodín. Presnosti systému s takto naladeným systémom sú uvedené v tabuľkách (tabuľky 9.2, 9.3, 9.4) nižšie.

Z tabuliek je zrejmé, že systém má najvyššiu presnosť pri použití posledných 30 dní dát. Toto platí pre všetky úseky a až na jednu výnimku pre všetky modely. ako najpresnejší sa opäť ukázal model **ClusterModel**, čím je potvrdený výsledok získaný v kapitole 8.

Vzhľadom na to, že bolo vybrané len jedno nastavenie modelov - to najlepšie z kapitoly 8, je možné, že existujú aj iné produkčné nastavenia (počty dní pre tréning), ktoré by priniesli požadované výsledky. Avšak toto hľadanie nastavení je časovo veľmi náročné, z toho dôvodu takýto test nebol vykonaný.

Aby bolo možné overiť presnosť systému v dostatočne krátkom čase, systém využíval umelú časovú škálu, teda bol maximálne zrýchlený. Výkon systému bol pri tomto teste ~300 predikčných cyklov za sekundu (jeden predikčný cyklus - všetky sekcie a modely).

Model	$MAPE_7$ [%]	$MAPE_{15}$ [%]	$MAPE_{30}$ [%]	$MAPE_{60}$ [%]
BCK	10.57	10.57	10.57	10.57
TimeDomainModel	14.34	11.05	10.28	31.20
ClusterModel	12.07	9.90	9.24	151.13
CombinedModel	57.83	10.85	10.28	24.17

Tab. 9.2: Presnosť predikčného systému pri produkčnom teste - úsek 1

Model	$MAPE_7$ [%]	$MAPE_{15}$ [%]	$MAPE_{30}$ [%]	$MAPE_{60}$ [%]
BCK	33.84	33.84	33.84	33.84
TimeDomainModel	38.73	21.33	10.74	284.85
ClusterModel	10.85	8.66	7.20	21.25
CombinedModel	37.03	16.11	36.47	402.54

Tab. 9.3: Presnosť predikčného systému pri produkčnom teste - úsek 2

Model	$MAPE_7$ [%]	$MAPE_{15}$ [%]	$MAPE_{30}$ [%]	$MAPE_{60}$ [%]
BCK	8.56	8.56	8.56	8.56
TimeDomainModel	8.60	8.01	7.81	8.04
ClusterModel	8.53	7.72	7.42	7.63
CombinedModel	10.72	8.97	8.00	8.47

Tab. 9.4: Presnosť predikčného systému pri produkčnom teste - úsek 3

10 NASTAVENIE PREDIKČNÉHO SYSTÉMU

Táto kapitola sa zaoberá nastavením predikčného systému a popisom inštalácie závislostí, ktoré sú potrebné pre beh systému. Tiež popisuje súbory, ktoré sú priložené k diplomovej práci a ich umiestnenie na priloženom médiu.

Nastavenia systému sú priamo naviazané na modul **configs**. Premenné a nastavenia sú zdokumentované v priloženej dokumentácii. V rámci nastavení je tiež potrebné nakonfigurovať úseky, pre ktoré má byť predikcia vykonaná (viď dokumentácia pre **section_settings**).

10.1 Spustenie systému

Samotné spustenie je možné dvomi spôsobmi - manuálne alebo automatizovane s využitím virtualizácie a kontajnerov - platformy Docker[1].

10.1.1 Spustenie s využitím platformy Docker

Spustenie prostredníctvom vopred pripraveného dockerového kontajneru - virtualizovaného systému - prináša viacero výhod. Zaručuje, že systém bude bežať vždy v tom istom prostredí, t.j. prostredí v ktorom bol otestovaný. Navyše týmto spôsobom je systém ľahko nasaditeľný na najrôznejšie platformy a nie je potrebné riešiť medziplatformnú kompatibilitu - napr. Linux vs. Windows, čo môže byť pri niektorých aplikáciách problematické.

Na spustenie prostredníctvom Dockeru je v prostredí Windows možné s využiť dávkový skript `run_training.bat` (tréning) a `run_prediction.bat` (predikcia). Týmto príde k automatickému stiahnutiu všetkých definovaných závislostí, prekopírovaniu kódu predikčného systému do novo vytvoreného obrazu systému a spusteniu dvoch instancií - kontajnerov tohto obrazu. Jedna instancia vykonáva predikciu a ďalšia je zodpovedná za vytvorenie (tréning) modelov.

10.1.2 Manuálne spustenie

Pri manuálnom spustení je možné využiť skript `run_system.py`, ktorým sú spustené predikčný aj tréningový modul zároveň, alebo nezávislým spustením tréningového (`run_training.py`) a predikčného (`run_prediction.py`) modulu.

Avšak pri nezávislom spustení je potrebné aby najprv prebehol tréning a až potom bol spustený predikčný modul, pretože inak bude systém bez platných modelov

[1] https://www.docker.com/what-docker

a bude produkovať neplatné predikcie. Pri spustení skriptu `run_system.py` je predikčný modul spustený s oneskorením dostatočným na to, aby vytvorenie modelov prebehlo.

Inštalácia závislostí

V rámci prípravy prostredia pre spustenie systému je potrebné nainštalovať určité závislosti, bez ktorých nie je možné systém spustiť.

Jedná sa najmä o interpreter jazyka Python (verzia 3.5.2) a nasledujúce moduly:

- `numpy` – 1.12.1
- `sklearn` – 0.18.1
- `scipy` – 0.19.0
- `pandas` – 0.19.2
- `pytz` – 2016.10
- `six` – 1.10.0
- `joblib` – 0.11
- `pymysql` – 0.7.10
- `matplotlib` – 1.5.1
- `seaborn` – 0.7.1

Väčšinu z týchto modulov obsahuje distribúcia `Anaconda` [29], prípadne je možné na inštaláciu Python-u a závislostí využiť dávkový skript `prepare_environment.bat`, ktorý je spolu s potrebnými súbormi v prílohe.

11 ZÁVER

Táto diplomová práca sa zaoberá problematikou predikcie dojazdových dôb. V rámci tejto témy boli zosumarizované významné práce venujúce sa tejto problematike, ako aj rôzne predikčné metódy strojového učenia - hlavne sa jedná o regresné a clusterizačné algoritmy.

Bola vypracovaná analýza reálnych dát z existujúceho definovaného úseku, dáta boli tiež vizualizované a na ich základe boli navrhnuté príznaky, ktoré by bolo možné použiť pri tvorbe predikčných modelov.

Boli navrhnuté tri predikčné modely, ktorých úspešnosť bola overená prostredníctvom offline aj produkčného testu. Ako najúspešnejší model bol vyhodnotený model **ClusterModel**, ktorý mal v produkčných testoch chybu MAPE na každom úseku o viac než 12 % menšiu ako súčasne používaný model.

V rámci projektu bol implementovaný testovací framework - súbor funkcií a tried pre vizualizáciu dojazdových dát, ich spracovanie a vyhodnocovanie úspešnosti modelov. Bol použitý na nájdenie vhodných parametrov modelov, spracovanie dát a ich vizualizáciu.

Implementovaný bol aj komplexný predikčný systém, ktorý má napojenia na existujúcu infraštruktúru firmy CAMEA a ktorého výstupom sú predikované hodnoty dojazdovej doby na definovaných úsekoch, ktoré sú dostupné v minútových intervaloch. Predikčný systém sa dokáže vysporiadať s čiastočnými výpadkami zdrojov dát.

Predikčný systém bol navrhnutý modulárne tak, aby ho bolo možné ďalej vylepšovať vývojom nových predikčných modelov či postupov na estimáciu ich parametrov. Taktiež je možné systém pozmeniť čo sa týka práce s chýbajúcimi dátami - momentálny model dáta dopĺňa na základe štatistických deskriptorov, prípadne extrapoluje predchádzajúce dojazdové doby.

Nezávisle na zadaní práce bola ako ďalšia funkcionalita pridaná implementácia systému v rámci platformy Docker, čo umožňuje ľahké nasadenie systému prostredníctvom definovaného virtualizovaného stroja - kontajneru, ktorého tvorba je plne automatizovaná prostredníctvom priložených skriptov.

V prípade budúceho nasadenia na veľký počet úsekov sa ako perspektívna možnosť naskytá vytvoriť systém postavený na multiprocessingu s využitím viacerých jadier procesoru. V tom prípade by bola záťaž lepšie rozložená na celú výpočtovú jednotku a taktiež by sa využil väčší potenciál dnešných procesorov. Momentálne moduly systému multiprocessing nevyužívajú, nakoľko to bol požiadavok firmy.

LITERATÚRA

[1] SORIGUERA MARTÍ, Francesc. *Travel Time Definitions* [online]. s. 43 [cit. 2016-10-13]. DOI: 10.1007/978-3-662-48858-4_2. Dostupné z: http://link.springer.com/10.1007/978-3-662-48858-4_2

[2] RICE, J. a E. VANZWET. A Simple and Effective Method for Predicting Travel Times on Freeways. *IEEE Transactions on Intelligent Transportation Systems* [online]. 2004, 5(3), 200-207 [cit. 2016-10-10]. DOI: 10.1109/TITS.2004.833765. ISSN 1524-9050. Dostupné z: http://ieeexplore.ieee.org/document/1331390/.

[3] WU, C.-H., J.-M. HO a D.T. LEE. Travel-Time Prediction With Support Vector Regression. *IEEE Transactions on Intelligent Transportation Systems* [online]. 2004, 5(4), 276-281 [cit. 2016-10-10]. DOI: 10.1109/TITS.2004.837813. ISSN 1524-9050. Dostupné z: http://ieeexplore.ieee.org/document/1364002/.

[4] INNAMAA, Satu. *Short-term prediction of traffic flow status for online driver information* [online]. Espoo, Finland: VTT, 2009 [cit. 2016-10-10]. ISBN 978-951-3873-417. Dostupné z: https://aaltodoc.aalto.fi/handle/123456789/4627.

[5] LUNDE, Emil Øien a Thomas WOLFF. *Time Prediction: A Comparison Study on a common Data Set.* Trondheim, 2015. Dostupné také z: https://brage.bibsys.no/xmlui/bitstream/id/401494/12828_FULLTEXT.pdf. Master's thesis. Norwegian University of Science and Technology. Vedoucí práce Anders Kofod-Petersen.

[6] ELHENAWY, Mohammed, Hao CHEN a Hesham A. RAKHA. Dynamic travel time prediction using data clustering and genetic programming. *Transportation Research Part C: Emerging Technologies* [online]. 2014, 42, 82-98 [cit. 2016-10-12]. DOI: 10.1016/j.trc.2014.02.016. ISSN 0968090x. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0968090X14000588

[7] CHU, Lianyu, OH, S., RECKER, Will. Adaptive Kalman filter based freeway travel time estimation. In: *84th TRB Annual Meeting* [online]. Washington DC. 2005 [cit. 2016-10-10]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.330.7905&rep=rep1&type=pdf.

[8] HASTIE, Trevor., Robert. TIBSHIRANI a J. H. FRIEDMAN. *The elements of statistical learning: data mining, inference, and prediction* [online]. 2nd

ed. New York, NY: Springer, c2009 [cit. 2016-10-12]. ISBN 978-038-7848-587. Dostupné z: http://statweb.stanford.edu/~tibs/ElemStatLearn/printings/ESceLII_print10.pdf

[9] ZOU, Hui a Trevor HASTIE. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* [online]. 2005, 67(2), 301-320 [cit. 2017-03-16]. DOI: 10.1111/j.1467-9868.2005.00503.x. ISSN 1369-7412. Dostupné z: http://doi.wiley.com/10.1111/j.1467-9868.2005.00503.x

[10] WEBB, A. R. a Keith D. COPSEY. *Statistical pattern recognition*. 3rd ed. Hoboken: Wiley, 2011.

[11] Support Vector Machines. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-10-11]. Dostupné z: https://cs.wikipedia.org/wiki/Support_vector_machines

[12] MITCHELL, Tom M. *Machine learning*. Boston: McGraw-Hill, c1997. ISBN 0-07-042807-7.

[13] Neuronová síť: Přenosové funkce. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-10-12]. Dostupné z: https://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5

[14] DUCH, Włodzisław a Norbert JANKOWSKI. Survey of neural transfer functions. In: *Neural Computing Surveys*. 2. 1999, s. 163-212. Dostupné také z: ftp://ftp.icsi.berkeley.edu/pub/ai/jagota/vol2_6.pdf

[15] In a feed-forward neural network (NN), which type of neurons (linear or log-sigmoidal) must be placed in the hidden layers for pattern recognition and for regression ? In: *ResearchGate* [online]. 2013 [cit. 2016-10-12]. Dostupné z: https://www.researchgate.net/post/In_a_feed-forward_neural_network_NN_which_type_of_neurons_linear_or_log-sigmoidal_must_be_placed_in_the_hidden_layers_for_pattern_recognition_and_for_regression

[16] Multi-Layer Neural Network. *UFLDL Tutorial* [online]. [cit. 2016-10-12]. Dostupné z: http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

[17] Bootstrap aggregating. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-11-03]. Dostupné z: https://en.wikipedia.org/wiki/Bootstrap_aggregating

[18] BISHOP, Christopher M. *Pattern recognition and machine learning*. New York: Springer, c2006. Information science and statistics. ISBN 0-387-31073-8.

[19] DBSCAN. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-10-23]. Dostupné z: `https://en.wikipedia.org/wiki/DBSCAN`

[20] CUKIERSKI, William. Metrics. *Kaggle* [online]. Kaggle Inc. [cit. 2016-10-24]. Dostupné z: `https://www.kaggle.com/wiki/Metrics`

[21] Coefficient of determination. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-17]. Dostupné z: `https://en.wikipedia.org/wiki/Coefficient_of_determination`

[22] QIAO, Wenxin, Ali HAGHANI a Masoud HAMEDI. A Nonparametric Model for Short-Term Travel Time Prediction Using Bluetooth Data. *Journal of Intelligent Transportation Systems* [online]. 2013, 17(2), 165-175 [cit. 2016-10-26]. DOI: 10.1080/15472450.2012.748555. ISSN 1547-2450. Dostupné z: `http://www.tandfonline.com/doi/abs/10.1080/15472450.2012.748555`

[23] Friedman test. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-10-31]. Dostupné z: `https://en.wikipedia.org/wiki/Friedman_test`

[24] YILDIRIMOGLU, Mehmet a Nikolas GEROLIMINIS. Experienced travel time prediction for congested freeways. *Transportation Research Part B: Methodological* [online]. 2013, 53, 45-63 [cit. 2016-10-31]. DOI: 10.1016/j.trb.2013.03.006. ISSN 01912615. Dostupné z: `http://linkinghub.elsevier.com/retrieve/pii/S0191261513000465`

[25] PEDREGOSA, F., G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT a E. DUCHESNAY. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 2011, 12, 2825–2830 [cit. 2016-12-11]. Dostupné z: `http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf`

[26] VAN DER WALT, Stefan, S Chris COLBERT a Gael VAROQUAUX. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering* [online]. 2011, 13(2), 22-30 [cit. 2016-12-11]. DOI: 10.1109/MCSE.2011.37. ISSN 1521-9615. Dostupné z: `http://ieeexplore.ieee.org/document/5725236/`

[27] MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In: Stéfan van der WALT a Jarrod MILLMAN, ed. *Proceedings of the 9th Python in Science Conference.* 2010, s. 51–56.

[28] HUNTER, John D. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* [online]. 2007, 9(3), 90-95 [cit. 2016-12-11]. DOI: 10.1109/MCSE.2007.55. ISSN 1521-9615. Dostupné z: `http://ieeexplore.ieee.org/document/4160265/`

[29] *Anaconda* [online]. [cit. 2017-03-20]. Dostupné z: `https://anaconda.org/`

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

SVR	Support Vector Regression – algoritmus strojového učenia
ANN	Artificial Neural Network – umelá neuronová sieť, algoritmus strojového učenia
kNN	k-Nearest Neighbour – metóda k-najbližších susedov, algoritmus strojového učenia
DBSCAN	Density-Based Spatial Clustering of Applications with Noise – algoritmus na vyhľadávanie zhlukov, založený na hustote priestorových dát so šumom
MAE	Mean Absolute Error – Priemerná absolútna chyba
MAPE	Mean Absolute Percentage Error – Priemerná absolútna percentuálna chyba
RMSE	Root Mean Squared Error – Priemerná kvadratická chyba
RMSLE	Root Mean Squared Logarithmic Error – Priemerná kvadratická logaritmická chyba
LOKRR	Local Online Kernel Ridge Regression – Lokálne zameraný algoritmus Ridge Regression s online tréningom
GMM	Gaussian Mixture Models – Zmiešané gaussovské modely
IQR	Inter Quartil Range – Rozdiel hodnôt, ktoré nadobúda tretí a prvý kvartil
UML	Unified Modelling Language – Zjednotený modelovací jazyk

PRÍLOHY

Súčasťou diplomovej práce je:

- Zdrojový kód
- Dokumentácia kódu

Obsahová štruktúra prílohy (relevantných súborov) je uvedená nižšie.

```
code
├── dependencies
│   ├── python
│   └── prepare_environment.bat ..... Príprava prostredia pre man. spustenie
├── docs
│   ├── Test_suite_documentation.bat ..... Dokumentácia test_suite
│   └── Travel_time_system_documentation.bat ..... Dokumentácia systému
├── traveltime_system ..... Zdrojové súbory systému
│   ├── test_suite
│   ├── traveltimes_prediction
│   ├── run_system.py ..... Skript pre spustenie celého systému
│   ├── run_training.py ..... Skript pre spustenie tréningu
│   └── run_predition.py ..... Skript pre spustenie predikcie
├── docker
│   ├── run_training.bat ..... Spustenie tréningu - docker
│   └── run_prediction.bat ..... Spustenie predikcie - docker
└── readme.README
```