ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

Operatívne riadenie železničnej dopravy využitím strojového učenia

Dizertačná práca

Kód 28360020213005

Študijný program: Aplikovaná informatika
Študijný odbor: Informatika
Pracovisko: Katedra softvérových technológii
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline
Školiteľ: doc. Ing. Emil Kršák, PhD.

Žilina, 2021 **Ing. Tomáš Kello**

Poďakovanie

Chcel by som sa poďakovať školiteľovi práce doc. Ing. Emilovi Kršákovi PhD. za cenné rady a pomoc pri vypracovávaní dizertačnej práce, Správe železníc a ŽSR za poskytnutie detailných záznamoch o premávke na železničných tratiach, AŽD Praha s.r.o. za ich sprostredkovanie a Slovenskému hydrometeorologickému ústavu za poskytnutie dát o počasí z meteorologickej stanice Čadca a Turzovka.

ABSTRAKT V ŠTÁTNOM JAZYKU

Kello, Tomáš: *Operatívne riadenie železničnej dopravy využitím strojového učenia*. [Dizertačná práca]. – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra softvérových technológii. – Školiteľ: doc. Ing. Emil Kršák PhD. – Stupeň odbornej kvalifikácie: doktorandský. – Žilina: FRI UNIZA, 2021. Počet strán 125

Dizertačná práca sa zaoberá problémom predikcie jazdnej doby vlakovej súpravy. Matematický model plánovanej jazdnej doby dosahuje úspešnosť 85,61% K predikcii sme využili 5 ročné záznamy o premávke železničnej dopravy. Pomocou umelej inteligencie a vstupných dát sme natrénovali model, ďalej sme úspešnosť zlepšovali pomocou selekcie dát a nastavovaním parametrov učiaceho algoritmu. Výsledná predikcia skutočnej jazdnej doby dosiahla úspešnosť až 94,38%. Reprezentuje to zlepšenie o 8,77%.

Kľúčové slová: Strojové učenie, Predikcia jazdnej doby vlakovej súpravy, Tensor Flow, ML.NET

ABSTRAKT V CUDZOM JAZYKU

Kello, Tomáš: *Railway traffic management using machine learning*. [Doctoral thesis]. – University of Žilina. Faculty of management science and informatics; Department of software technologies. – Supervisor: doc. Ing. Emil Kršák PhD. – Professional qualification level: PhD. – Žilina: FRI UNIZA, 2021. Page count: 125

Topic of this work is based on prediction train driving time. Mathematical model has success rate of 85,61%. We used train driving records of 5 years interval for machine learning algorithm. To improve our success rate, we combine machine learning with selection of data and modifying training parameters. Our final prediction gets 94,38% success rate. This represents improvement of 8,77%.

Key words: Machine learning, Prediction of train driving time, Tensor Flow, ML.NET

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam rovníc

Zoznam skratiek

IS – Informačný systém

ISOŘ – Informačný systém operatívneho riadenia

GTN – dispečerský monitoring a riadenie železničnej dopravy s elektronickou dopravnou dokumentáciou

MS SQL – Microsoft SQL

SQL - Structured Query Language

SHMU – Slovenský hydrometeorologický ústav

ASVC - Automatické stavanie vlakových ciest

EF – Entity Framework

ML.NET – Machine Learning .NET

LINQ – Language Integrated Query

Ciele dizertačnej práce

Hlavným cieľom dizertačnej práce je vytvoriť modul umelej inteligencie s čo najpresnejších odhadom jazdnej doby vlakovej súpravy. K dosiahnutiu cieľa sme si stanovili nasledovné tézy, podľa ktorých budeme vo výskume postupovať:

Téza 1: Navrhnúť, vytvoriť a overiť model umelej inteligencie k odhadu jazdnej doby vlakovej súpravy na základe historických dát o jazde vlaku.

Téza 2: Navrhnúť algoritmus pre zvýšenie presnosti vypočítanej teoretickej jazdnej doby v medzistaničnom úseku využitím aktuálne dosiahnutých jazdných dôb daného vlaku v predchádzajúcich úsekoch.

Téza 3: Aplikovať algoritmus pre zvýšenie presnosti (výsledok tézy 2) na vytvorený model umelej inteligencie (výsledok tézy 1) k získaniu čo najlepších odhadov.

Téza 4: Overiť, či je možné na základe jazdných dôb spätne odhadovať parametre vlakovej súpravy.

Základným krokom k splneniu očakávaných cieľov je získanie historických dát o jazdných dobách vlakov. K tomu sme si vybrali oblasť Českej republiky vďaka jej hustote premávky a oblasť Slovenskej republiky vďaka prístupu k dátam o počasí. V Českej oblasti s 23 stanicami sa pokúsime získať čo najviac záznamov, čo nám umožní vytvoriť presnejšie modely umelej inteligencie. V prípade Slovenskej oblasti sa zameriame na počasie a jeho vplyvy na jazdnú dobu.

Ďalším krokom bude vytvoriť základný predikčný model. S ktorým budeme ďalej experimentovať so snahou zníženia chyby. Taktiež využiť viaceré technológie a implementácie.

Bod 3 a 4 opisujú neštandardné spôsoby, od ktorých očakávame zlepšenie výsledkov. Sú silno viazané na druh dát ktoré získame. Vďaka aktuálnym informáciám o vlakovej súprave a jej aktuálneho správania môžeme presnejšie predikovať jej budúce správanie.

Očakávaný prínos

Pre reálne využitie tejto práce v praxi by sme chceli vytvoriť softvérový systém, ktorý bude obsahovať implementovaný model predikcie reálnej jazdnej doby. Tento bude zakomponovaný do aktuálneho dispečerského riadiaceho systému vlakovej dopravy, aby dispečer resp. automatizovaný riadiaci systém dostal presnejšiu informáciu o príchode vlaku do stanice, čo výrazne prispeje k zvýšeniu presnosti pri určovaní momentu optimálneho postavenia jazdnej cesty pre vlak na vstupe do stanice. To vedie k celkovému zvýšeniu priepustnosti dopravy a optimalizácie využitia koľají a jednotlivých prvkov vlakových ciest.

Druhou aplikáciu je presnejšia lokalizácia vlakovej súpravy a riešenie vlakových priecestí, kde dispečer resp. automatizovaný riadiaci systém môže manipulovať s priecestím a podržať ho zavreté dlhšie po prechode prvého vlaku, príp. zatvoriť ho vo vhodnom okamžiku, aby celková priepustnosť priecestia bola maximálna a aby sa eliminovali prípady, kedy je nutné spomaliť vlak lebo by musel čakať pred priecestím kým sa uvoľní.

Úvod

Žijeme v dobe, v ktorej nové dáta vznikajú exponenciálnou rýchlosťou. Zo štatistík vyplýva, že v čase písania tejto práce pribudne každý deň viac ako 2,5 miliardy GB nových dát a toto číslo neustále rastie. [1] Množstvo dát, ktoré máme k dispozícii je enormné, a preto je potrebné vedieť efektívne pracovať s dátami a využiť ich v náš prospech. Jednou z možnosti, ako využiť dáta, je strojové učenie, ktoré patrí do podoblasti umelej inteligencie. Inteligentné systémy, ktoré používajú algoritmy strojového učenia dokážu predikovať výsledky na základe predchádzajúcich dát, pričom vedia objaviť súvislosti a prepojenia v dátach. Využitie strojového učenia je v dnešnej dobe veľmi široké a môžeme sa s ním stretnúť v oblastiach ako zdravotníctvo, právo, marketing a v rôznych iných odvetviach. Spolu s umelou inteligenciou máme efektívny nástroj na spracovanie veľkého množstva dát, ktoré môžu byť v textovom, zvukovom, alebo vo vizuálnom formáte.

V posledných rokoch sme sa stali svedkami rozvoju mnoho nových frameworkov na vývoj modelov strojového učenia. Keďže je strojové učenie relatívne nový odbor vedy, nedá sa objektívne usúdiť, ktorý jazyk, resp. framework, je na integráciu strojového učenia do aplikácií najlepší. „Boom“ v oblasti sa dá pripísať technológiám s otvoreným zdrojovým kódom a sprístupneniu vývojových nástrojov širokej verejnosti. Spoločnosť Microsoft za posledných pätnásť rokov investovala veľa prostriedkov do snahy umelej inteligencie a výsledkom je multiplatformový framework s otvoreným zdrojovým kódom známy pod názvom ML.NET. Medzi ďalších aktívnych členov v oblasti umelej inteligencie patrí GOOGLE s frameworkom Tensor-Flow a cloudovým riešením Google Cloud Platform. Medzi konkurenciu v oblasti cloudových riešení patrí tiež Microsoft s ich Microsoft Azure službou, Amazon s AWS (Amazon Web Service), IBM a ďalší.

Riešeným problémom je nepresnosť doterajšieho matematického modelu, ktorý počíta teoretickú jazdnú dobu vlaku na známych úsekoch. Matematický model využíva detailné informácie o trati ale nezahŕňa informácie o stave, opotrebovaní trate a ďalšie ťažko definovateľné veličiny. Preto sa pokúsime tento výpočet zlepšiť s použitím historických dát o jazdných dobách na daných úsekoch, tie nepriamo v sebe obsahujú všetky aktuálne problémy, ktoré sa na trati vyskytujú.

Na záver práce využijeme všetky získané znalosti a vytvoríme službu, ktorá bude spustená na webovej adrese. Tým sa použiteľnosť práce zvýši a integrácia v praxi bude

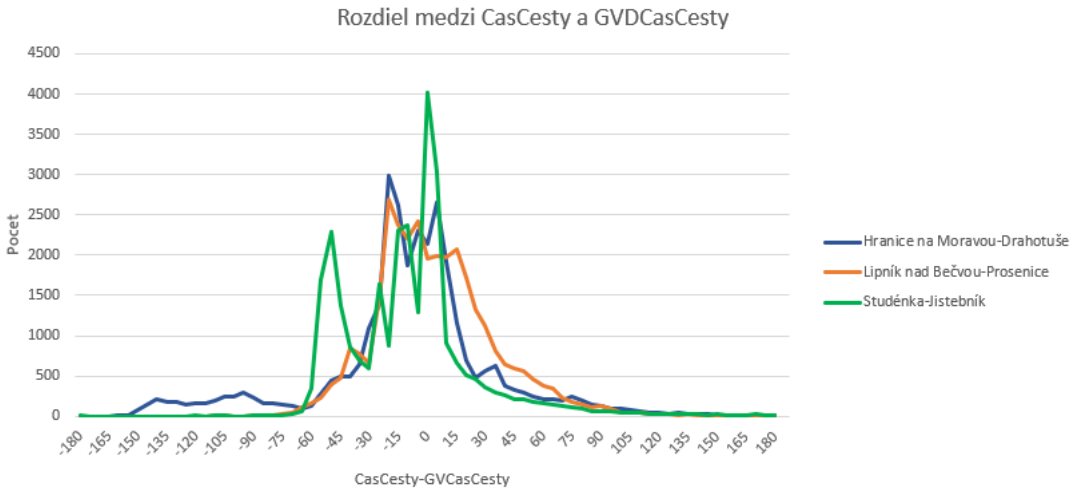
jednoduchšia. V závere taktiež zhodnotíme dosiahnuté výsledky spolu s výhodami a nevýhodami jednotlivých metód k zlepšeniu predikcie umelej inteligencie.

1 Stav riešenej problematiky

Na Slovensku aj v Českej republike sa v dnešnej dobe používajú na určenie jazdnej doby vlaku matematické modely.

Výpočet jazdnej doby vlaku v matematických modeloch je realizovaný pomocou výpočtov z parametrov vlaku a trati. Parametre jazdy vlaku medzi dvoma stanicami sa popisujú z dvoch pohľadov, a to kinematického a dynamického. Konkrétne parametre kinematického pohľadu sú napríklad rozbehnutie, jazda v stabilnej rýchlosti a spomaľovanie. Z dynamického pohľadu je to jazda silou, výbeh a brzdenie. [2]

Tieto modely, ktoré produkujú teoretické výsledky, neberú do úvahy typ a stav lokomotívy, historické záznamy, námrazu na koľajniciach a iné faktory, ktoré môžu ovplyvňovať jazdnú dobu vlaku. Z toho dôvodu vznikajú meškania. Ako môžeme vidieť na obrázku 1, skutočná jazdná doba (CasCesty) sa vo veľa prípadoch nezhoduje s teoretickou jazdnou dobou (GVDCasCesty) a ich rozdelenie hustoty je sústredené v intervale -60s (skrátil jazdnú dobu) a +30s (zvýšil meškanie). Môžeme teda usúdiť, že v realite sa vyskytuje veľa faktorov, ktoré teoretický model do úvahy neberie.

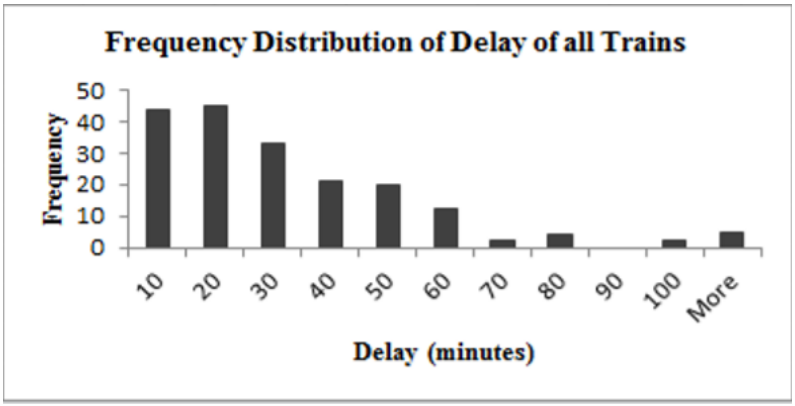


Obrázok 1: Graf rozdielu času medzi skutočnou a teoretickou hodnotou.

S cieľom minimalizovať meškanie vlakov sa vo svete začali dopravné spoločnosti, ako aj výskumníci, zaujímať o možnosti tvorby nových, presnejších modelov. S veľkým vzostupom strojového učenia vo svete informačných technológií sa táto možnosť stala jasnou voľbou. Vďaka tomu, že dopravné spoločnosti vlastnia veľké súbory dát, presne také, aké pre strojové učenie potrebujeme, má strojové učenie v sektore dopravy veľký potenciál. Túto skutočnosť si začali všímať aj zahraničné inštitúcie.

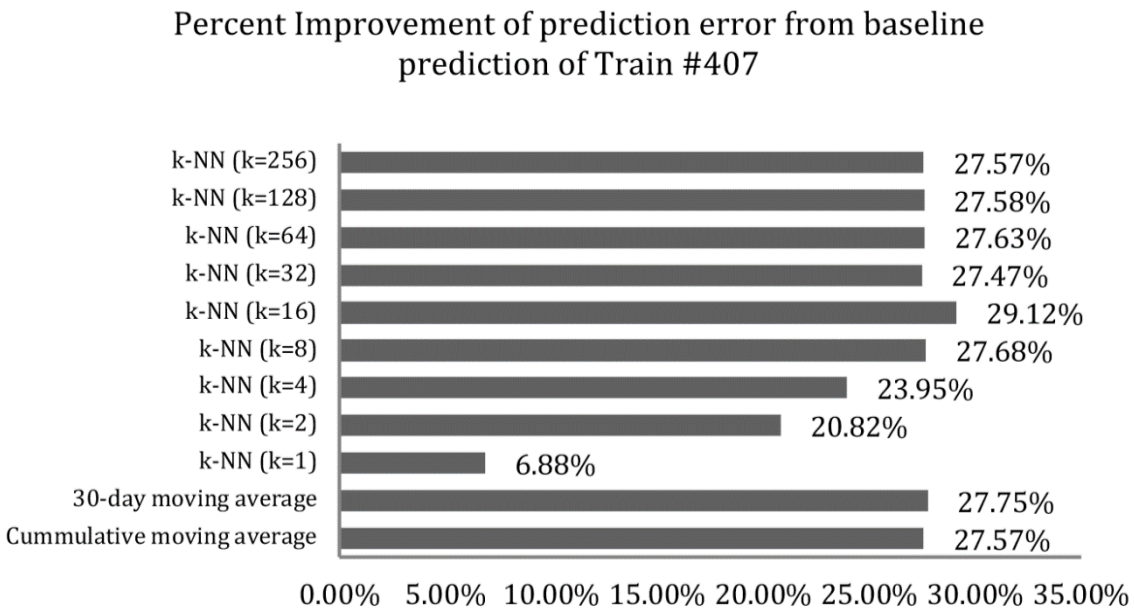
1.1 Vylepšenie predikcie meškania v Thajsku

V roku 2014 vydalo Thajské národné centrum elektroniky a počítačových technológií vedeckú publikáciu, v ktorej skúmali, či sa dá zlepšiť predikcia meškania vlaku pomocou predikčného modelu, ktorý berie do úvahy historické záznamy o čase cesty. Výskum uskutočnili na trati štátnej vlakovej dopravnej spoločnosti. V Thajsku cestujúcich denne prevážа 160 až 180 vlakov, čo predstavuje 2.2 až 3.3 milióna cestujúcich mesačne. Železnice tu bojujú s veľkým problémom, keďže väčšina vlakov mešká ako môžeme vidieť na obrázku 2, ktorý je z roku 2013, kedy vlak v období jún-november meškal priemerne 23 minút a doprava sa väčšinou riadila operatívne výrazne neefektívne. Od vtedy boli trate v Thajsku zlepšené, ale stále sa to nedá porovnať so Slovenskou alebo Českou železničnou dopravou, kde dodržiavanie grafikonu a minimalizácia meškania je prioritou. Dáta na výskum získavali šesť mesiacov zo stránky dopravnej spoločnosti, ktorá podáva informácie o meškaní manuálne zadávané dopravcom. Po dobu šesť mesiacov sťahoval systém pravidelne každú minútu zdrojový kód HTML zo spomínanej stránky dopravnej spoločnosti. [3]



Obrázok 2: Meškanie vlakov v Thajsku, jún-november 2013 [2]

V experimente použili na tvorbu predikčného modelu dva algoritmy, algoritmus založený na kĺzavých súčtoch a algoritmus založený na k-najbližšom susedovi. Napriek malému množstvu vstupných dát, ktoré mali dostupné zo súboru, sa podarilo dosiahnuť zlepšenie odhadu jazdnej doby oproti aktuálne používanému matematickému modelu. Jednu z predikcií, ktorú sa podarilo zlepšiť, môžeme vidieť na obrázku 3. [3]



Obrázok 3: Zlepšenie predikcie v Thajsku [2]

1.2 Vylepšenie predikcie meškania v Číne

Dňa 4 februára 2019 bola v Číne vydaná vedecká publikácia s názvom Train delay analysis and prediction based on big datafusion. Publikácia sa týkala analýzy a predikcie meškania vlakov v Číne. [4]

Čínski kolegovia na rozdiel od thajských zahrnuli do modelu navyše premennú počasie. Dáta o vlakoch zbierali tri mesiace začiatkom roka 2018. Dáta o počasí z 344 miest boli získané z Čínskej meteorologickej dátovej siete. Databáza o počasí obsahovala teplotu, smer a silu vetra a v neposlednom rade typ počasia. Pomocou grafickej reprezentácie a výpočtu korelácie medzi počasím a časom jazdy vlaku usúdili, že počasie má nezanedbateľný vplyv na jazdnú dobu vlakov. Kvôli bezpečnosti musia vlaky v horšom počasí ísť pomaly. No v prípade počasia, ktoré môžeme definovať ako normálne, spozorovali, že väčší vplyv na presnosť predikcie jazdnej doby majú historické záznamy, než počasie samotné.

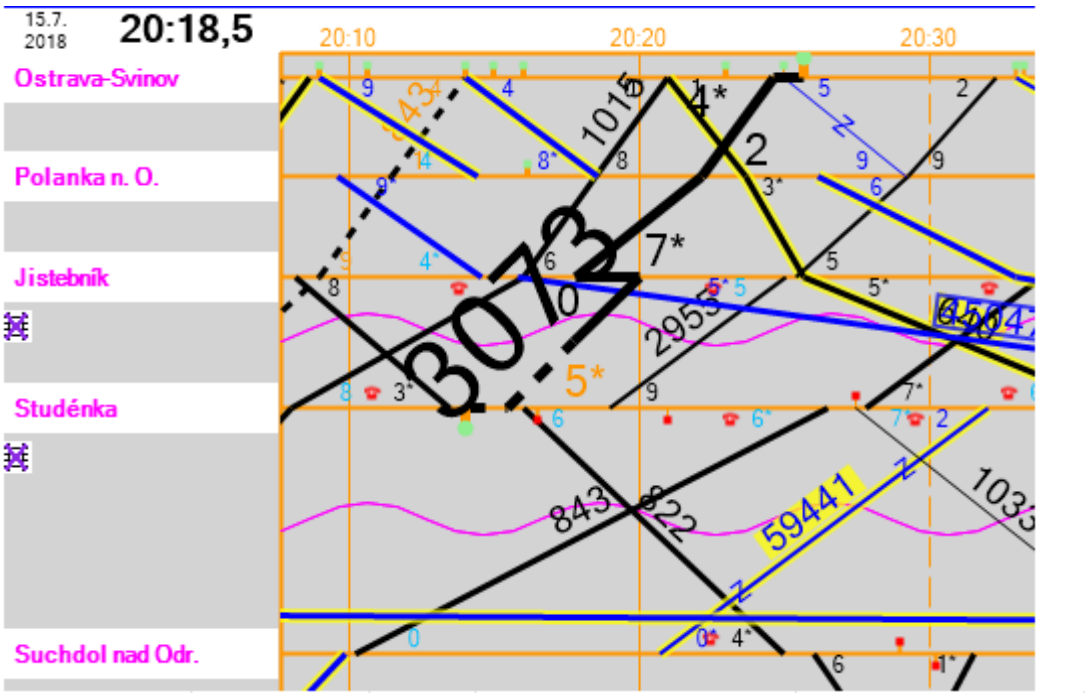
Predikčný model bol založený na modeli „boosted decision trees“ (zosilnené rozhodovacie stromy). Výsledkom bol predikčný model, ktorý dokázal spoľahlivo predikovať meškanie vlaku, žiaľ s väčším počtom chýb, ako by bolo pre prax akceptovateľné. [4]

1.3 Aktuálne riadenie železničnej oblasti

K riadeniu a zobrazovaniu aktuálnej situácie v riadených oblastiach využívajú dispečeri v Českej republike a na Slovensku informačný a riadiaci systém GTN. Ten zobrazuje takzvanú plachtu, teda pohyby jednotlivých vlakových súprav v čase a na trati. Zároveň podporuje zobrazenie železničnej trate spolu s koľajami jednotlivých staníc.

Obrázok 4 Riadiacie stredisko

Na obrázku 4 je zobrazený náhľad do riadiaceho strediska, kde na pozadí môžeme vidieť aktuálny stav dopravnej siete, pozíciu vlakov, obsadenosť a rezerváciu koľají.



Obrázok 5 GTN zobrazenie plachty

Ďalšie dôležité zobrazenie je pozícia vlakov v čase, ktoré je zobrazené na obrázku 5. Na Xovej osi v pohľade z ľava do prava plynie čas. Jednotlivé čiary reprezentujú vlaky, ktoré sa s časom presúvajú medzi stanicami. Stanice sú na Y osi reprezentované horizontálnou čiarou. Počas prevádzky je zobrazená ešte vertikálna čiara reprezentujúca aktuálny čas. Tá rozdeľuje graf na ľavú časť reprezentujúcu históriu a pravú časť, reprezentujúcu plánovanú prevádzku v riadenej oblasti. V prípade, že stanica obsahuje viac koľaji, je možné si ju rozkliknúť a pozrieť, na ktorú koľaj má daný vlak plánovaný príchod, prípadne kam prišiel.

2 Strojové učenie

Umelá inteligencia a strojové učenie sú dva od seba neoddeliteľné pojmy, ale neznamenajú to isté. Strojové učenie možno definovať ako aplikáciu umelej inteligencie v praxi, s cieľom učiť počítač z dát. Cieľom je učenie počítača na konkrétnu úlohu za účelom maximalizovania výkonu počítača pre túto úlohu, pomocou spracovania informácií z dát.

Strojové učenie má za sebou dlhú históriu a vďaka matematikom a počítačovým vedcom a postupným vývojom výpočtovej techniky je dnes súčasťou našich životov. Medzi známe míľniky strojového učenia sú roky:

- 1952 – vytvorenie programu, ktorý zlepšoval svoju hru v dáme čím viac ťahov odohral,
- 1979 – študenti zostrojili vozidlo, ktoré vedelo samo plánovať cestu a vyhýbať sa prekážkam,
- 1996 – počítač od IBM s názvom Deep Blue porazil v šachu šachového veľmajstra,
- 2009 – tím *BellKor's* Pragmatic Chaos vyhral súťaž od Netflixu na vytvorenie najlepšieho algoritmu na odporúčanie filmov,
- 2011 – počítač IBM Watson vyhral vedomostnú súťaž Jeopardy!,
- 2012 – neurónová sieť od Google dokáže rozpoznávať ľudí a mačky vo videách na portáli YouTube,
- 2016 – program AlphaGo porazil v stolnej hre Go svetového šampióna,
- 2017 – *Aplphabet*'s Jigsaw tím vytvoril systém, ktorý dokáže identifikovať nevhodné príspevky na webových stránkach. [5]

Strojové učenie spolu s umelou inteligenciou sú považované za veľké výdobytky dnešného sveta a pomáhajú nám žiť zdravší a produktívnejší život. Veľmi často sa strojové učenie používa na rozpoznávanie obrázkov. Algoritmus dokáže vyhľadať na obrázkoch konkrétne objekty a alebo tváre. Pri rozpoznávaní reči dokáže software rozpoznať slová, ktoré sú hovorené v audio formáte a následne ich konvertovať do textového súboru. V zdravotníctve dokáže diagnostikovať choroby. Klasifikačný model na základe klinických parametrov určuje, či pacient má predpoklady dostať danú chorobu. Porozumením záznamom o kriminalite môžeme predpovedať, kde a kedy sa

očakáva ďalší zločin a predísť mu. Polícia Los Angeles sa snaží predpovedať zločin používaním algoritmov, ktoré využívajú údaje o doteraz spáchaných zločinoch a s určitou presnosťou môžu určiť oblasť a čas v ktorej nastane ďalší zločin. Okrem polície aj spoločnosti ako Google, Netflix, Spotify a mnoho ďalších používajú algoritmy založené na strojovom učení k skvalitneniu svojich služieb, ktoré poskytujú svojim zákazníkom. [6]

Nevýhody strojového učenia:

- Nutnosť kvalitných dát – strojové učenie vyžaduje veľké množstvo záznamov na ktorých sa algoritmus natrénuje, v prípade nekvalitných dát ťažko môžeme očakávať kvalitné výsledky
- Čas a zdroje – v prípade sofistikovaných algoritmov je potrebný výpočtový výkon a čas, čo môže viesť k dodatočným finančným nákladom
- Interpretácia výsledkov – výsledky poskytnuté algoritmom je potrebné správne vyhodnotiť a interpretovať
- Náchylnosť na chyby – algoritmus sa učí samostatne bez zásahu človeka, čo môže viesť k chybám. Ak založíme naše rozhodovanie na chybných výsledkoch algoritmu, takéto rozhodnutie môže prispieť k dodatočným problémom
- Nezaručená chyba – po vytvorení modelu umelej inteligencie, nemôžeme presne definovať, akú bude mať model najväčšiu chybu
- Komplikované ladenie – ak sa natrénovaný algoritmus rozhodne pre nejaký výsledok, je veľmi komplikované spätne zistiť detailne prečo sa tak rozhodol.

Rozdelenie algoritmov strojového učenia:

1. Učenie s učiteľom
2. Učenie bez učiteľa
3. Kombinované učenie – kombinácia prvých dvoch spomenutých
4. Spätnoväzobné učenie

2.1 Využitie strojového učenia

Nárast dôležitosti strojového učenia je pevne prepojený s nárastom v oblasti „big data“. Práve pomocou týchto dát je možné použiť strojové učenie ako techniku na riešenie problémov v rôznych odvetviach, akými sú napríklad:

- Výpočtové financie – pre algoritmické obchodovanie na burze cenných papierov, kryptomeny.
- Spracovávanie obrázkov a počítačové videnie (image processing and computer vision) – rozpoznávanie tváre, pohybu, a rozpoznávanie objektov.
- Výpočtová biológia – zachytenie rakoviny v skorých štádiách, sekvencovanie DNA, výskum nových liekov.
- Produkcia energie – zníženie ceny energie a predpovedanie záťaže.
- Automobilový, letecký a výrobný priemysel - na predpovedanie zlyhania, opotrebenia, autonómne vozidlá.
- Prirodzený spôsob spracovávania reči – hlasoví asistenti.
- Personalizované online reklamy, odporúčanie filmov, hudby – napr. v prípade Amazonu, Netfixu, Spotify.
- Finančné podvody – systémy na zachytenie podozrivých aktivít, resp. transakcií na účtoch.

2.2 Kedy použiť strojové učenie

Algoritmy strojového učenia nachádzajú skryté závislosti (vzťahy) v dátach, ktoré nám sprostredkujú lepší prehľad pri rozhodnutiach a predpovediach. Používajú sa každodenne ako pomocný nástroj pri rozhodnutiach v oblasti zdravotnej starostlivosti, obchodovania na burze, produkcie energie, atď. Konkrétne využitie nástrojov strojového učenia je napríklad odporučenie filmu alebo seriálu na základe preferencií predplatiteľa služby a preferencií podobných používateľov. Ďalej môžeme uviesť, že predajcovia sa pomocou tohto nástroja snažia získať informácie o správaní svojich zákazníkov.

Riešiť problém pomocou strojového učenia je vhodné, keď máme k dispozícii veľký objem dát zahrňujúci veľký počet premenných, ale žiadny vzorec alebo rovnicu popisujúcu správanie dát. Nasledujúce situácie sú vhodnými príkladmi situácií, ktoré je vhodné riešiť za pomoci strojového učenia:

- Vytvorenie modelu na základe funkcií a rovníc je príliš zložité na implementáciu, napr. rozpoznávanie tváre a ľudskej reči.
- Pravidlá sa v systéme neustále menia – rozpoznávanie podvodov podľa záznamov transakcií.

- Charakter dát sa sústavne mení a program sa musí prispôsobiť – automatizované obchodovanie, predpoveď spotreby energie, obchodných trendov.

2.3 Učenie s učiteľom

Pri učení s učiteľom, alebo so vzorom je cieľom vytvoriť funkciu, ktorá najlepšie odhaduje výstupnú premennú *y*. Musíme si uvedomiť, že učiteľom, alebo vzorom je práve premenná *y*, podľa ktorej sa vytvára funkcia, ktorá ju opisuje. Učenie prebieha na trénovacích dátach, ktoré obsahujú výstupnú premennú *y* a vstupné premenné *x*.

Môže sa stať, že nastane jav pretrénovanie (overfitting). V tom prípade je algoritmus príliš naviazaný na špecifickosť testovacej množiny, nevie generalizovať a funkcia príliš presne opisuje testovacie dáta. Tento jav vzniká pri nedostatku dát v trénovacej množine, pri veľkom množstve vstupných premenných *x* alebo pri nepresných vstupných dátach. Opakom je podtrénovanie (undrefitting), kedy algoritmu unikli spojenia a dôležité súvislosti v dátach a tým nie je schopný pracovať správne. [7]

Ďalej ho môžeme rozdeliť na dve podoblasti:

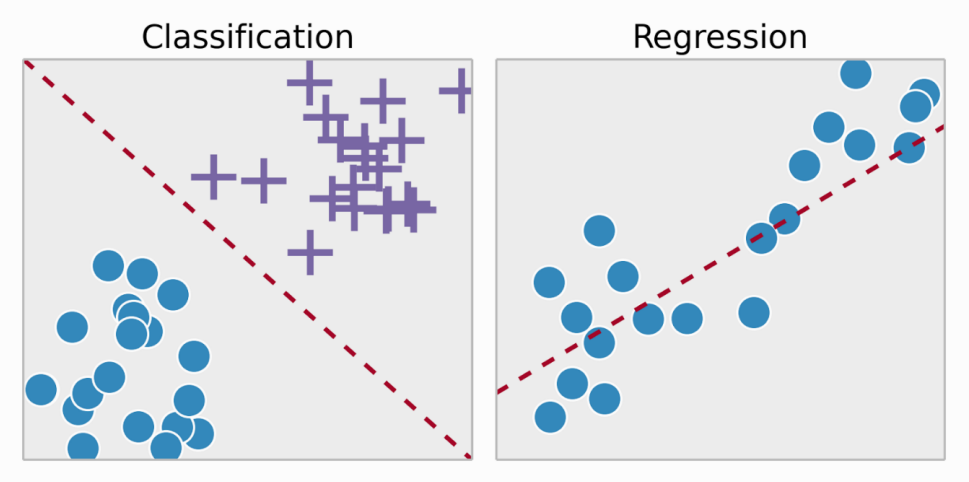
- Regresia,
- Klasifikácia.

2.3.1 Regresia

Regresný algoritmus mapuje funkciu f , kde vstupom sú nezávisle premenné *x* na závislú spojitú výstupnú premennú *y*. Z toho vyplýva, že jej definičný obor je reálne číslo.

2.3.2 Klasifikácia

Klasifikačný algoritmus mapuje funkciu f , kde vstupom sú nezávisle premenné *x* na závislú výstupnú premennú *y*, ktorá môže nadobúdať iba diskrétne hodnoty. Ak závislá premenná môže nadobúdať iba dva rôzne výstupy, tak sa jedná o binárnu klasifikáciu a ak ich je viac ako dva tak je to viactriedna klasifikácia. Na obrázku 6 vidíme rozdiel medzi klasifikáciou a regresiou.



Obrázok 6: Klasifikácia a regresia

Príklady algoritmov

- Lineárna regresia,
- Logistická regresia,
- Rozhodovacie stromy,
- Náhodný les,
- Algoritmus k-najbližších susedov.

Lineárna regresia

Matematický vzťah pre lineárnu regresiu: $y = a + b \times x$ .V tomto prípade máme jednu nezávislú premennú, v praxi sa však vyskytujú problémy, ktoré obsahujú viac nezávislých premenných, vtedy používame model viacnásobnej lineárnej regresie. Jeho matematický vzťah môžeme vidieť na obrázku 1. V tomto vzťahu „y“ reprezentuje závislú premennú, „a“ reprezentuje priesečník regresnej priamky s osou y, „b_i“ koeficient, ktorý určuje sklon regresnej priamky , „x_i“ reprezentuje nezávislú premennú, „n“ počet nezávislých premenných v modeli a „ε“ reprezentuje náhodnú chybu.

$$\boldsymbol{y = a + (\sum_{i=1}^{n} b_i \times x_i) + \varepsilon}$$

Rovnica 1: Lineárna regresia

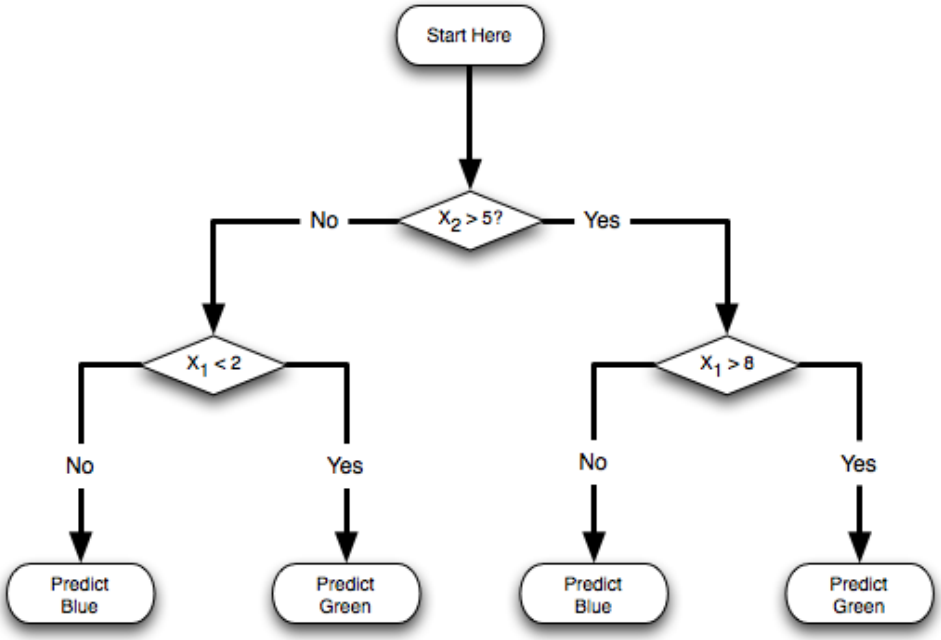
Pomocou techniky minimalizácie chyby sa snažíme získať najlepšiu regresnú priamku pre naše dáta. Termín chyba v tejto terminológií značí rozdiel medzi očakávanou

a odhadnutou hodnotou výstupu. Na výpočet chyby môžeme použiť viaceré možnosti, ktoré budú opísané v odstavci 2.7

Rozhodovacie stromy

Ako z názvu tejto metódy vyplýva, rozhodovacie stromy vyzerajú podobne ako stromový diagram. Strom začína svojim koreňom, a od neho algoritmus vyberá smer k ďalšiemu vrcholu. Vrcholy reprezentujú rozhodnutia alebo udalosti, ktoré sa navzájom neovplyvňujú a nemôžu nastať naraz.

Rozhodovacie stromy si môžeme predstaviť ako otázky, ktoré predikčný model kladie, aby čo najpresnejšie určil predikciu. Na obrázku 7 môžeme vidieť diagram rozhodovacích stromov a otázky, ktoré sa kladú pri každom vrchole.



Obrázok 7: Diagram modelu rozhodovacieho stromu

Regresné rozhodovacie stromy

Regresné rozhodovacie stromy sú rozhodovacie stromy, ktoré pracujú so spojitým výstupom. Snahou je predikovať hodnotu výstupnej numerickej premennej na základe vstupných premenných, pomocou kombinácie rozhodovacích stromov a lineárnej regresie. Tvorba stromov je zabezpečená pomocou rekurzívneho delenia. Regresné stromy väčšinou používajú algoritmus CART, celým názvom Klasifikačné a regresné stromy. Výber najlepšieho rozdelenia v každom vrchole stromu je riadený pomocou minimalizácie chyby.

Neurónové siete

Ďalším typom metódy strojového učenia s učiteľom sú neurónové siete. Neurónové siete sú navrhnuté podľa princípu fungovania ľudského mozgu. V neurónových sieťach sa v každej vrstve model učí iné charakteristiky očakávanej výstupnej premennej, na podobnom princípe pracuje aj mozog. Spracovanie informácie, s ktorou sa človek stretáva, zabezpečuje viacero častí mozgu naraz. [8]

Neurónové siete sa väčšinou rozdeľujú na priamo-väzbové (FNN) a rekurentné neurónové siete (RNN).

2.4 Učenie bez učiteľa

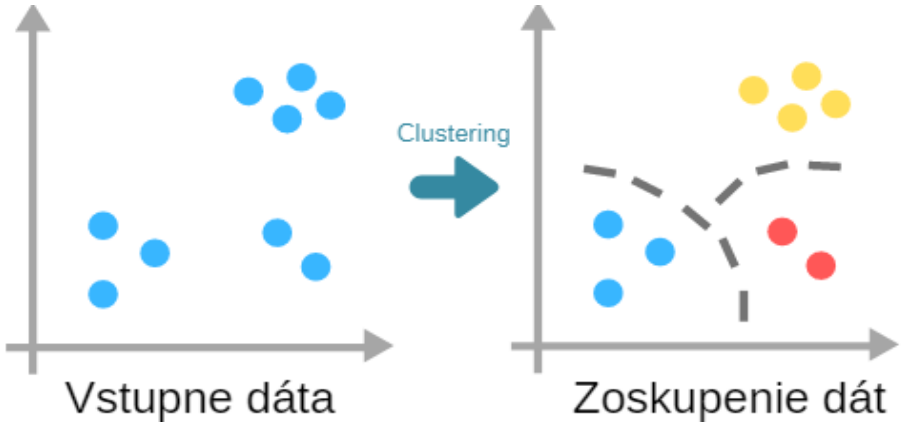
Pri tomto type nemáme vzor, to znamená, že už nemáme výstupnú premennú *y*. Dôsledkom toho nie je fáza trénovania, podľa ktorej by sa algoritmus naučil mapovať vstupné premenné *x* na výstupnú premennú *y*. Algoritmy tohto typu sami hľadajú a objavujú súvislosti v dátach. Snažia sa získať pridanú hodnotu z dát, alebo objaviť anomálie, ktoré by ľudia sami nevedeli nájsť. Tento typ je užitočný aj v prípade, keď nie je jasné čo hľadať v dátach. [9]

Príklady algoritmov učenia bez učiteľa:

- Zhlukovanie,
- Detekcia anomálii
- Metóda hlavných komponentov

2.4.1 Zhlukovanie

Používa súbor dát, v ktorom sú známe iba vstupy a hľadá v nich štruktúru pomocou zhlukovania. Metóda zhlukovania je metóda, ktorá sa snaží zoskupovať elementy do podobných kategórií na základe ich podobných vlastností.



Obrázok 8: Zhlukovanie

V tomto prípade môžeme modelu povedať, koľko skupín chceme mať na výstupe, prípadne maximálnu možnú odlišnosť dát v skupine. [10]

2.5 Spätnoväzobné učenie

V prípade tohto typu strojového učenia sa model učí v interaktívnom prostredí metódou pokusu a omylu. Na model sa v rámci učenia spätnou väzbou pozeráme ako na agenta pôsobiaceho v určitom prostredí. Agent sa časom učí ovplyvňovaním svojho prostredia a následným pozeraním sa na dôsledky interakcie.

Aj keď učenie pod dohľadom a učenie spätnou väzbou sa pokúšajú mapovať vstupy na výstupy, učenie spätnou väzbou na rozdiel od učenia pod dohľadom posilňuje model využitím odmien a trestov, ktoré predstavujú signály pre žiadane a nežiadane správanie.

Pre použitie tejto metódy. Je potrebné definovať funkciu, ktorou získame ohodnotenie každého stavu. Tá sa využije na výpočet odmien a program dokáže sám hľadať najlepšie riešenie ďalšieho kroku. Pre predstavu môžeme použiť šach, kde sa program snaží poraziť súpera. V tomto prípade je potrebné definovať koniec hry. Ak sa dostane vlastné kráľ z hry von, stav musí mať veľmi nízke ohodnotenie, keďže to znamená koniec hry a prehru programu. V opačnom prípade a prechodu do stavu vyhodenia súperovho kráľa musí mať tento ťah vysoké ohodnotenie, aby sa tento krok stal cieľom programu.

2.6 Korelácia a Asociácia

Vzťah medzi dvoma veličinami vieme zhodnotiť ich vizualizáciou do grafu alebo pomocou korelačných koeficientov.

Dôležitým pojmom v korelačnej analýze je kovariancia. Reprezentuje asociáciu medzi dvoma premennými. Problém s kovarianciou nastáva pri porovnávaní dvoch kovariancií, pretože nie je vyjadrená v univerzálnych jednotkách, ale v jednotkách získaných pomocou stredných hodnôt a distribučnej funkcie daných premenných. Normovaním kovariancie pomocou meracej sústavy, ktorá meria variabilitu v dátach získame koreláciu. Korelácia produkuje hodnoty z konzistentnej škály. [11]

Koreláciu vyjadrujú korelačné koeficienty. Jedným z týchto koeficientov je Pearsonov korelačný koeficient, používa sa na vyčíslenie lineárneho vzťahu medzi dvoma veličinami a nadobúda hodnoty od – 1 do +1. Podľa potreby môžeme použiť aj iné často využívané korelačné testy, Spearmanov alebo Kendallov koeficient. [11]

Vzťah medzi dvoma kategorickými premennými nazývame asociácia. V prípade že chceme zistiť asociáciu medzi dvoma premennými používame asociačné koeficienty, ako napríklad Cramérov V koeficient.

2.6.1 Pearsonov korelačný koeficient

Pearsonov koeficient je najčastejšie používaný koeficient, aj napriek tomu, že bol definovaný pred 120 rokmi Karlom Pearsonom. Často používaný matematický vzťah Pearsonovho koeficientu je zobrazený na rovnici 2, kde „σ_X, σ_Y“ vyjadrujú štandardnú odchýlku a „μ_X, μ_Y“ vyjadrujú priemery premenných „X“ a „Y“. Čitateľ je identický s definíciou kovariancie. [12]

$$\rho_{X,Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

Rovnica 2: Pearsonov korelačný koeficient

Pomocou delenia kovariancie štandardnými odchýlkami premenných „X“ a „Y“ zabezpečíme, že korelačný koeficient bude nadobúdať hodnoty z intervalu $< -1,1 >$.

2.6.2 Cramérov V koeficient

Pre zistenie asociácie medzi dvoma kategorickými premennými môžeme použiť Pearsonov Chi-kvadrát test, ale ten iba vyjadrí či asociácia medzi danými kategorickými premennými existuje. Preto použijeme Cramérov V koeficient, aby sme vedeli určiť ako silná asociácia to je. Výpočet koeficientu je zobrazený na rovnici 3, kde „χ^2“ reprezentuje hodnotu Chí-kvadrátu, „n“ je počet pozorovaní v Chi-kvadrát teste, „k“ počet kategórií v jednej premennej a „r“ počet kategórií v druhej premennej. Koeficient nadobúda hodnoty z intervalu od < 0,1 >. [13]

$$V = \sqrt{\frac{\chi^2}{n * \min(k-1, r-1)}}$$

Rovnica 3: Cramérov V koeficient

2.7 Výpočet chyby a porovnanie algoritmov strojového učenia

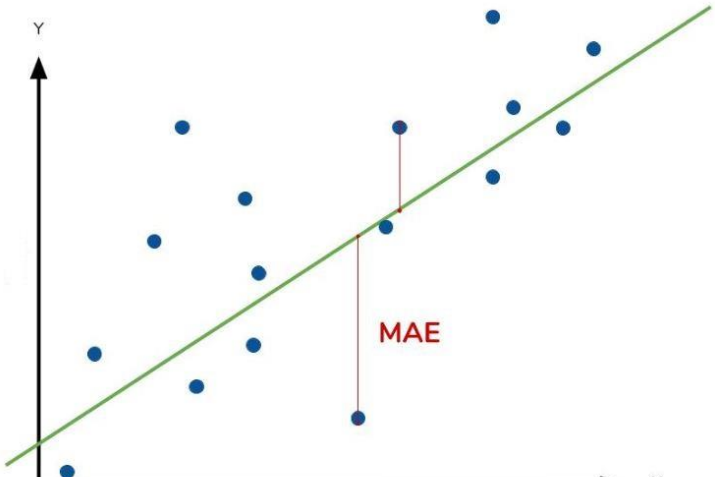
Po odhade parametrov a získaní predpovedaných výsledkov z regresného algoritmu je potrebné vedieť, ako dobre algoritmus predpovedá výstupnú premennú *y*, keď ju porovnáme so skutočnou hodnotou na testovacej množine dát. Rozdiel medzi skutočnými hodnotami a predpovedanými hodnotami nazývame rezíduami (odchýlky). Pre všetky dátové položky môžeme vypočítať ich rezídua, ktoré sa použijú vo výslednej metrike. Na základe metriky môžeme posúdiť užitočnosť a presnosť predikcie algoritmu. Medzi niektoré metriky hodnotenia regresných algoritmov patria: [14]

- MAE
- MAPE
- MSE
- RMSE
- R^2

2.7.1 MAE Mean Avarage Error

MAE, alebo priemerná absolútna chyba je najjednoduchšia regresná metrika k pochopeniu. Vypočíta sa ako suma absolútnych hodnôt rezíduí pre každú dátovú položku vydelená počtom dátových položiek, ako môžeme vidieť [15]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5}$$



Obrázok 6: Priemerná absolútna chyba [18]

Na obrázku 6 môžeme vidieť grafické zobrazenie MAE. Zelená čiara je regresná priamka algoritmu a modré bodky predstavujú pôvodné dátové položky. Môžeme vidieť rozdiel medzi predikovanými a skutočnými hodnotami. Väčšie aj menšie chyby prispievajú lineárne k celkovej sume chýb. Ak je MAE malé číslo, tak máme veľmi dobrý model, naopak veľké hodnoty naznačujú, že model je nekvalitný. Ak je MAE = 0, tak to znemená, že algoritmus dokonale presne predpovedá výstupné hodnoty, ale v praxi málo pravdepodobné. [15]

2.7.2 **MAPE Mean Avarage Percentage Error**

MAPE, alebo priemerná absolútna percentuálna chyba je podobná MAE . Vypočíta sa ako suma absolútnych hodnôt rezíduí, kde každé rezíduum je vydelené pôvodnou hodnotou a vynásobením číslom 100 a vydelením počtom dátových položiek získame percentuálny údaj:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{6}$$

Keďže sa jedná o percentuálny údaj, tak MAPE metrika je ľahko pochopiteľna a interpretovateľná. Túto metriku nemôžeme použiť pre nulové dátové položky, lebo by sme delili nulou. [15]

2.7.3 MSE Mean Square Error

MSE je suma všetkých rozdielov medzi očakávanou a predikovanou hodnotou výstupnej premennej, delená počtom dát. Z tejto formulácie vyplýva, že MSE trestá veľké chyby veľmi prísne. Tento matematický vzťah je vyobrazený na rovnici 4, kde „y_i“ reprezentuje očakávanú hodnotu a „$\hat{y}_i$“ reprezentuje hodnotu predikcie. [16]

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

Rovnica 4: MSE

2.7.4 RMSE Root Mean Squared Error

Predstavuje druhú odmocninu z priemernej hodnoty reziduálneho súčtu štvorcov (MSE). Rezíduá, resp. reziduálne odchýlky, sú rozdiely medzi skutočnými a odhadovanými hodnotami. Reziduálnu odchýlku i-teho pozorovania značíme e_i:

$$e_i = y_i - \hat{y}_i$$

Rozdiel môže byť buď kladný alebo záporný, preto je potrebné ho umocniť. Predpokladajme, že y_i je skutočná hodnota i-teho pozorovania a $\hat{y}_i$ je vypočítaný odhad, potom sa ukazovateľ počítaný z n pozorovaní dá matematicky vyjadriť nasledovne:

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

2.7.5 R2 Koeficient determinácie

Druhou štatistikou vypovedajúcou o kvalite modelu je koeficient determinácie (Coefficient of determination, známy aj ako R-squared). Nadobúda hodnoty z intervalu od 0 po 1. Informuje o tom ako dobre regresné odhady aproximujú skutočné hodnoty, teda či sú odhady sústreďované okolo regresnej krivky. Formálnejšie sa dá koeficient definovať ako percento z celkového rozptylu závislej premennej y spôsobené vplyvom regresie na nezávislú premennú x. Ak je hodnota ukazovateľa rovná 1, tak model perfektne odhaduje skutočné hodnoty. V opačnom prípade hodnota bližšia ku 0 značí, že

priemer závislej premennej aproximuje závislú premennú lepšie, alebo rovnako dobre ako regresný model. Model teda nie je možné napasovať na vzorku validačných dát. Koeficient je možné vypočítať nasledovne:

$$R^2 = \frac{SS_{reg}}{SS_{tot}} = 1 - \frac{SS_{res}}{SS_{tot}}, \text{ kde:}$$

SStot – celkový súčet štvorcov odchýlok dát od priemeru závislej premennej (total sum of squares), opisuje celkovú variabilitu závislej premennej y.

$$SS_{tot} = \sum_i (y_i - \bar{y})^2$$

$$SS_{res} + SS_{reg} = SS_{tot}$$

SSreg – regresný súčet štvorcov odchýlok predikcií od priemeru závislej premennej (regression sum of squares), opisuje variabilitu závislej premennej y spôsobenú vplyvom regresie.

$$SS_{reg} = \sum_i (\hat{y}_i - \bar{y})^2$$

SSres – reziduálny súčet štvorcov (residual sum of squares), MSE * n, opisuje variabilitu závislej premennej y, ktorá nie je spôsobená vplyvom regresie.

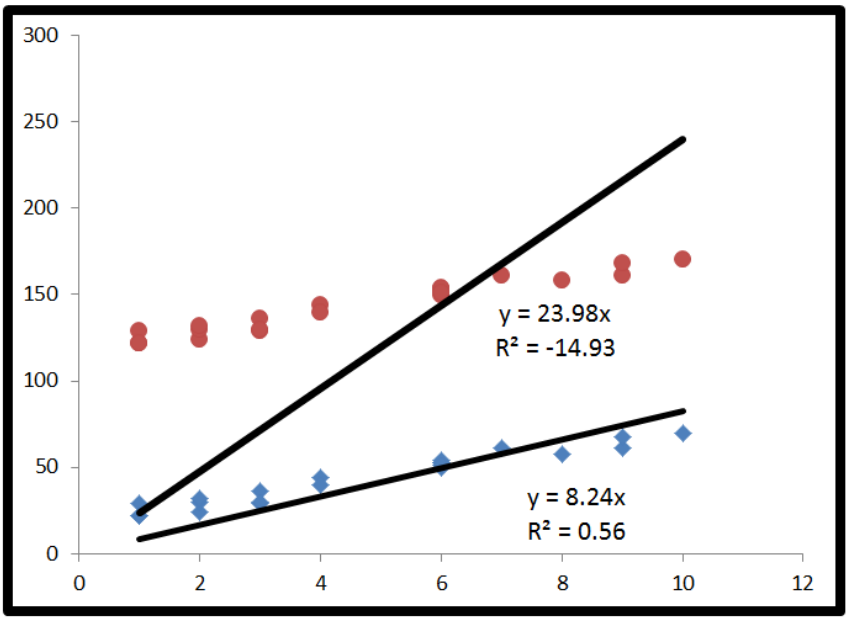
$$SS_{res} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i e_i^2 = MSE * n$$

$\bar{y}$ – priemer z odsledovaných vzoriek.

$$\bar{y} = \frac{1}{n}\sum_i y_i$$

2.7.6 Špeciálne prípady

Koeficient R^2 môže nadobúdať záporné hodnoty, ak model vyhovuje dátam horšie ako základný model, ako môžeme vidieť na obrázku 9.



Obrázok 9: Graf zobrazujúci model lineárnej regresie so záporným R^2

Ak dátam perfektne vyhovuje základný model, teda ak očakávané hodnoty sa rovnajú predikovaným hodnotám základného modelu, v tom prípade R^2 bude nedefinovateľné. Nastane totiž situácia zobrazená v rovnici 5. Keďže delenie nulou v aritmetike nie je definované, R^2 nebude možné vypočítať. [15]

$$R^2 = 1 - \frac{SSE}{0}$$

Rovnica 5 Delenie nulou

2.8 Postup trénovania a testovania

Trénovanie modelu úzko závisí na zvolenom algoritme. V prípade lineárnej regresie postačuje jeden priechod vstupnými dátami a zostavenie parametrov. V prípade neurónových sieti používame parameter „Epochs“ reprezentujúci počet opakovaní trénovania. Pri nízkej hodnote sa modelu nepodarí dostatočne natrénovať a odhaliť závislosti v dátach. V prípade vysokej hodnoty sa model pretrénuje.

V prípade testovania je potrebné oddeliť dáta ešte pred trénovaním tak, aby sa model testoval na neznámych dátach, ktoré neboli použité pri trénovaní. Chyba predikcie sa v takom prípade určí na testovacej množine. Druhou možnosťou je použitie krížovej validácie, kde sa toto rozdelenie spraví viac krát a tým sa každý záznam dostane raz do testovacej množiny a zvyšné razy je použitý v trénovanej fáze. V našom prípade

nemôžeme použiť náhodné rozdelenie, keďže jazdná súprava s unikátnym číslom obsahuje niekoľko záznamov vo vstupnej množine. Opisujúce jednotlivé časti trate medzi stanicami.

Aby sme predišli situácii, že záznam z koncovej stanici spolu s výsledným meškaním sa objavil v trénovanej množine a predikovali by sme predchádzajúci úsek trate daného vlaku, rozdeľovali sme dataset na základe id vlaku. To znamená, že všetky záznamy konkrétneho vlaku musia patriť jednej skupine. Buď trénovanej alebo testovacej. [9]

3 Možnosti implementácie

Vďaka veľkej popularite umelej inteligencie sa v oblasti informačných technológii objavilo niekoľko možností implementácie. Jedným z dôležitým rozdielom je miesto výpočtového uzla. Jednou z možností je beh na lokálnom počítači. V tomto prípade vzniká problém s výkonom. Keďže umelá inteligencia vyžaduje veľa zdrojov počas fáze učenia, ukazuje sa lepšie riešenie v podobe klaudov. Takéto riešenie prináša väčšina gigantických spoločností v oblasti výpočtovej techniky. Patria tam Google s Google Cloud Platform, Microsoft so službou Microsoft Azure, Amazon s Amazon Web Services (AWS), IBM a ďalší. Okrem výhody zapožičania si hardvéru a platby len za to, čo naozaj zákazník využije ponúkajú niektoré z nich grafické rozhranie, ktoré čiastočne obmedzuje funkcionalitu ale rapídne zrýchľuje vývoj a jednoduchosť.

Medzi frameworky, používané na vývoj modelov umelej inteligencie patrí Googlom vyvíjaný TensorFlow, Microsoftom ML.NET,

3.1 Tensorflow

Tensorflow je open source framework od spoločnosti Google, ktorý uľahčuje všetky procesy tvorby predikčného modelu, od získavania dát až po predikciu. Hlavné zameranie tohto framework-u sú neurónové siete, má zabudované mnohé algoritmy a modely pre prácu s nimi, avšak zvláda aj prácu s jednoduchšími modelmi, ako napríklad modely lineárnej regresie. Na pozadí využíva vysokovýkonné aplikácie naprogramované v C++, ale API má pre Python. Doterajšia podpora Tensorflow API pre C++ je slabšia (2019). [17]

3.1.1 Spôsob fungovania

Vývojári používajúci Tensorflow majú možnosť vytvárať štruktúry, ktoré popisujú, ako dáta prechádzajú grafmi, teda postupnosť spracovávania uzlov v grafoch.

Každý uzol v grafe je reprezentáciou matematickej operácie, a pod „tensorom“ rozumieme viacdimenzionálne dátové pole, reprezentujúce každé spojenie medzi uzlami. [17]

3.1.2 Výhody

Veľkou výhodou Tensorflow-u je vysoký level abstrakcie, ktorý ponúka. Táto vlastnosť predstavuje pre vývojára veľké uľahčenie tvorby modelov. Vývojár sa nemusí sústrediť na implementáciu algoritmov. To všetko zabezpečuje framework na pozadí, a tým pádom sa môže zameriavať len na celkovú logiku aplikácie.

Tensorflow taktiež ponúka možnosť pre vývojárov nahliadnuť do vnútra framework-u, napríklad framework ponúka mód, ktorý môže vývojár ohodnotiť a upraviť každú operáciu grafu zvlášť.

Ďalšou veľkou výhodou Tensorflow-u je možnosť využitia implementácie natrénovaného modelu do rôznych platforiem, napríklad do prehliadačov pomocou JavaScriptu, alebo do mobilov a vstavaných systémov, pomocou kompresie cez Tensorflow Lite Converter.

Najnovšou novinkou je možnosť naprogramovania Tensorflow aplikácie pomocou jazyku Swift, čo je aktuálne v beta verzii (2019), čím sa otvára veľký priestor pre využitie v systémoch macOS, iOS, watchOS, tvOS, atď.

3.1.3 Možné implementácie v programovacích jazykoch

Ako sme už vyššie spomenuli, hlavné Tensorflow API je pre jazyk Python, avšak Tensorflow ponúka aplikačné rozhrania aj pre iné jazyky. Pre tieto jazyky nemá Tensorflow plnú podporu algoritmov a funkcionality.

C

Jediný jazyk, ktorý projekt Tensorflow označil za jazyk s plnou podporou pre Tensorflow API. Mnoho programovacích jazykov má mechanizmy prepojenia sa s jazykom C a tým prepojiť TensorFlow aj s inými jazykmi za cenu obmedzenia konverzie dátových typov medzi nimi.

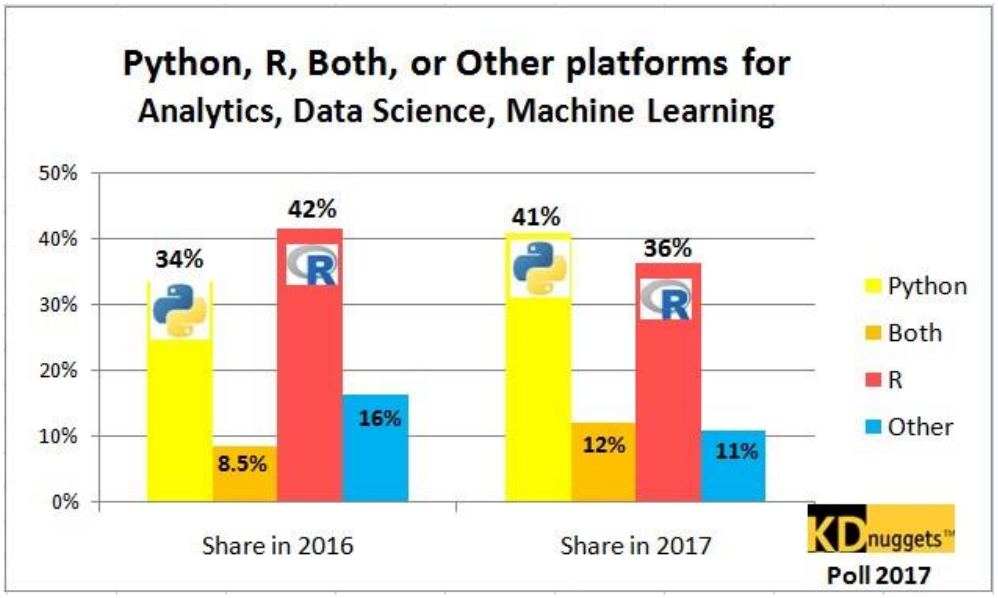
Python

Pre Python má Tensorflow skoro plnú podporu. Python bol prvým jazykom, v ktorom bolo sprístupnené API. Obsahuje skoro všetky funkcie, ktoré Tensorflow ponúka. Dôvodom pre výber Python-u ako hlavného jazyka pre Tensorlfow bola jednoduchosť implementácie framework-u pre väčšinu dátových analytikov a expertov na strojové učenie. Ďalším dôvodom bol veľký počet knižníc pre predspracovanie dát,

ktoré Python obsahuje. Príkladmi knižníc, ktoré Python obsahuje pre spracovanie dát sú NumPy, SciPy, pandas, Scikit-learn, a matplotlib.

R

Jazyk R, bol niekedy kráľom medzi jazykmi, keď sa jednalo o prácu s dátami. Podľa KDnuggets bol jazyk R v roku 2016 najpoužívanejším jazykom v oblasti týkajúcej sa práce s dátami. Ako ale môžeme vidieť na obrázku 10, v roku 2017 bol porazený vyššie spomenutým jazykom Python. [18]



Obrázok 10: Porovnanie využitia programovacích jazykov na strojové učenie

To však neznamená, že jazyk R nie je stále silným hráčom na trhu, a z tohto dôvodu má podporu aj pre Tensorflow API. Firma RStudio ale zvolila iný prístup, ako je odporúčaný od projektu TensorFlow, a teda namiesto využitia Tensorflow API pre jazyk C na prepojenie framework-u s jazykom, sa rozhodli pre obalenie Python API. R API tak dáva používateľom plnú funkcionalitu Python API.

C++

V tomto jazyku sú, ako sme už vyššie spomenuli, napísané Tensorflow aplikácie, ktoré bežia na pozadí. Aj napriek tomu je C++ API stále považované za experimentálne a má prístup k menej funkciám ako Python API.

Ďalšie jazyky

Tensorflow podporuje aj niektoré menej používané jazyky v oblasti strojového učenia. Napríklad programovací jazyk Go od spoločnosti Google, od ktorej je aj samotný

Tensorflow. Jazyk Java, veľmi silne zastúpený jazyk na trhu programovacích jazykov, a ako alternatívu k Jave aj jazyk C#. Taktiež niektoré ďalšie menej známe jazyky. Všetky tieto spomenuté jazyky majú iba čiastočnú podporu Tensorflow API.

3.2 ML.NET

ML.NET je framework strojového učenia podporujúci modelovo orientované strojové učenie vyvinuté pre .NET developerov. Umožňuje vývojárom využiť znalosti programovacích jazykov C# a F# na ľahké integrovanie prispôsobených modelov strojového učenia do existujúcich aplikácií bez potreby podrobných znalostí vo vývoji a ladení modelov strojového učenia. Môže mať taktiež využitie v akademickom prostredí, či byť použitý ako výskumný nástroj. Výhoda tohto frameworku je, že vývojári nemusia mať rozsiahle vedomosti z oblasti vedy o dátach. Framework je postavený na .NET štandarde, tým pádom zdedil schopnosť spustiť model na rôznych platformách. Framework ML.NET je považovaný za relatívne nový, aj keď sa začal prvý krát využívať v roku 2002 pod menom TMSN (text mining search and navigation) na interné účely spoločnosti Microsoft. [10]

Entity Framework

Slúži k práci s databázou. Umožňuje pracovať s dátami, ako by ich obsahoval samotný program a počas jeho behu sa sťahujú len tie, ktoré sú potrebné. Zabezpečujú teda komunikáciu medzi databázou a programom a tým výrazne urýchľuje vývoj a prehľadnosť kódu

Modelovo-orientovaný prístup

Framework dovoľuje zapracovať znalosti z domény modelovania systémov a vytvoriť algoritmus učiaci sa z dát na základe prispôsobeného modelu.

Open source a cross-platform

Ml.net predstavuje softvér s otvoreným zdrojovým kódom a je možné ho spustiť na operačných systémoch Windows, Linux, a macOS.

Osvedčený a rozšíriteľný

Vývojári majú možnosť využiť ten istý framework, ktorý bol použitý pri vývoji známych funkcií, akými sú napríklad Windows Hello, Bing Ads, a PowerPoint design ideas pri vylepšovaní vlastných aplikácií. ML.NET zahŕňa knižnice strojového učenia vytvorené spoločnosťou Microsoft Research. Vďaka rozšíriteľnosti frameworku bude možné postupom času využiť ďalšie populárne knižnice strojového učenia, napríklad Accord.NET, CNTK, TensorFlow a Scikit-learn.

Prispôsobenie modelov

ML.NET sa používa na vývoj a integrovanie prispôsobených modelov strojového učenia do .Net aplikácií rôznych typov – web, mobil, desktop, hranie a IoT.

Interpretovateľnosť

Podstatným problémom aplikácií strojového učenia je efekt čiernej skrinky, kedy osoba tvoriaca model nevie povedať s istotou prečo algoritmus v určitom momente preukazuje dané správanie, prípadne či sú dáta skreslené, zaujaté.

Ak máme mnoho premenných, ktoré ovplyvňujú výsledné skóre, tak ML.NET dokáže rozčleniť premenné a znaky, ktoré mali na výslednú kvalitu modelu najväčší vplyv. Ďalej počas trénovania a odstraňovania chýb modelu si môžeme prezrieť prefiltrované dáta.

Strojové učenie v prehliadači

ML.NET umožňuje užívateľom exportovať natrénované modely do ONNX (Open Neural Network Exchange) formátu. Tým pádom sa tieto modely môžu použiť aj v iných prostrediach, ktoré nie sú súčasťou ML.NET. Modely by bolo možné následne spustiť v prehliadači použitím ONNX.js, čo je javaskriptový framework klientskej strany pre modely strojového učenia uložených v ONNX formáte.

Ďalšie vlastnosti

ML.NET spája do kopy načítavanie dát, transformáciu dát a trénovanie modelu do jedného procesu. Dáta je nutné premeniť do formátu a typu potrebného pre spracovanie vstupov. Transformácie definované v modeli sa aplikujú ako na dáta na trénovanie, tak na vstupné dáta určené pre vytvorenie odhadov s natrénovaným modelom.

Dáta do modelu je možné dostať načítaním z niektorého podporovaného zdroja:

1. Text (CSV, TSV)
2. Parquet
3. Binary
4. IEnumerable<T>

Ostatné funkcie poskytované frameworkom:

- Textové transformácie.
- Ošetrenie chýbajúcich záznamov v dátach.
- Kategorické zakódovanie premenných.
- Normalizácia.
- Výber relevantných vlastností trénovania.

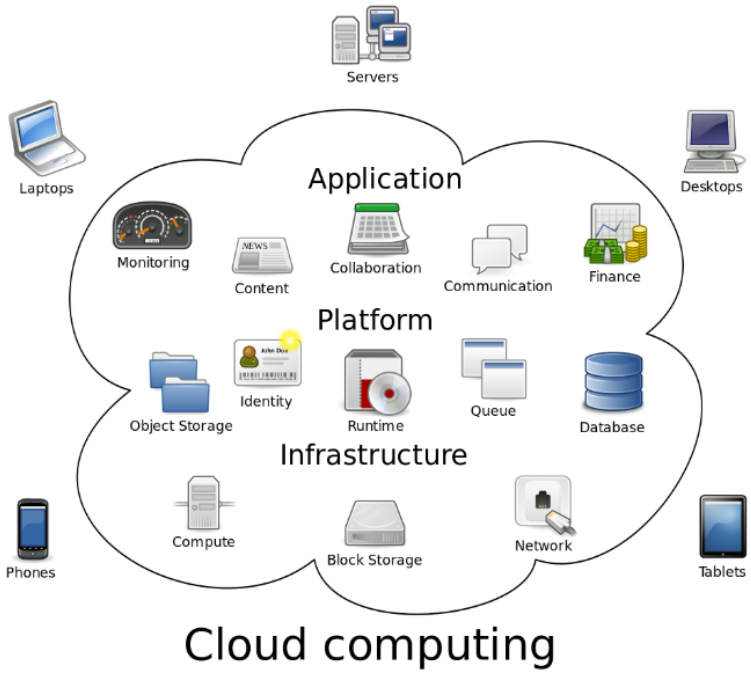
3.3 Cloud computing

V dnešnej dobe je bežné zálohovať svoje dáta bezplatne na cloudovom internetovom úložisku. Schopnosť ukladať dáta odkiaľkoľvek a z akéhokoľvek zariadenia ak máme internetové pripojenie je iba jedna z vlastností cloudov. Kým cloud je iba samotné úložisko na internete, cloud computing je technológia umožňujúca používateľom získať prístup k údajom, aplikáciám a službám cez internet. Nie je potrebný výkonný hardvér, ale stačí webový prehliadač a internetové pripojenie, pretože všetky služby sú uložené na serveri. Na cloudoch môžeme nájsť rozličné typy softvéru od rôznych kancelárskych balíkoch, až po zložité algoritmy, ktoré využívajú umelú inteligenciu. Možnosti cloudov sú široké a neustále rastie počet ich užívateľov. [19]

Nie všetky cloudy sú rovnaké a existuje viacero modelov a typov služieb, ktoré pomáhajú poskytnúť vhodné riešenia. Podľa typu implementácie cloudu môžeme cloudové služby rozdeliť do troch kategórií: [14]

- **Verejný cloud** je určený pre širokú verejnosť. Poskytovateľmi sú napríklad Google, Amazon alebo Microsoft. Poskytovatelia dodávajú svoju výpočtovú techniku, za ktorú si používateľ platí. K službám sa pristupuje cez internet, prostredníctvom webového prehliadača.
- **Privátny cloud** funguje výlučne vo vyhradenej infraštruktúre alebo dátovom centre. Ponúka rovnaké výhody ako verejný cloud a je vyhradený pre jedného nájomcu. Nie je zdieľaný s inými organizáciami a je bezpečnejší.

- **Hybridný cloud** kombinuje verejný a privátny cloud. Spoločnosť môže uchovávať citlivé údaje na privátnom cloude a zároveň využívať služby verejného cloudu. Hybridný cloud spája výhody verejného a privátneho cloudu.



Obrázok 11: Znázornenie fungovania cloud computingu

3.4 Google Cloud platform

Cloudové riešenie od giganta Google ponúka svoje služby firmám a zákazníkom po celom svete. Cloudové služby, ktoré poskytuje zahŕňajú oblasti ako web hosting, umelá inteligencia, strojové učenie, dátová analytika, databázové systémy, poskytovanie výpočtového výkonu a mnoho ďalších služieb. Medzi nástroje umelej inteligencie a strojového učenia patrí extrahovanie metadát z videí, prevod hlasu na text, dynamický preklad medzi rôznymi jazykmi, spracovanie neštruktúrovaného textu atď. V tejto bakalárskej práci sme využili nástroje Cloud Datalab, Google BigQuery a programovací jazyk Python, ktorý sa používal v prostredí Cloud Datalab. [20]

3.4.1 Cloud Datalab

Cloud Datalab je výkonný, interaktívny nastroj, vytvorený na skúmanie, analýzu, transformáciu, vizualizáciu údajov a vytváranie modelov strojového učenia na cloude

Google Cloud Platform. Pred používaním Datalabu je potrebne vytvoriť si v prostredí Googlu Cloudu takzvaný virtuálny stroj o určitom výpočtovom výkone na ktorom Datalab vykonáva svoju prácu. Ďalej je možne v prostredí Datalabu využiť Google BigQuery, ktorý implementuje SQL databázu a uľahčuje prácu s dátami.

Na písanie scriptov v Datalabe sa využíva programovací jazyk Python a softvérové knižnice pandas a scikit-learn. Knižnica pandas umožňuje vytvárať dátové objekty z SQL tabuliek, alebo CSV súborov a reprezentuje ich tabuľkou. Ďalej umožňuje manipuláciu, filtráciu, spájanie, rozdeľovanie, mazanie a vkladanie údajov v rámci dátového objektu. Knižnica scikit-learn špeciálne slúži na dátovú analýzu a dolovanie dát. Poskytuje algoritmy strojového učenia pre klasifikáciu, regresiu, zhlukovanie a ďalej podporuje spracovanie textu, validačné metódy a matematickú štatistickú metódu PCA. Táto knižnica podporuje aj metriky hodnotenia modelov, ktoré sú implementované v balíku metrics.

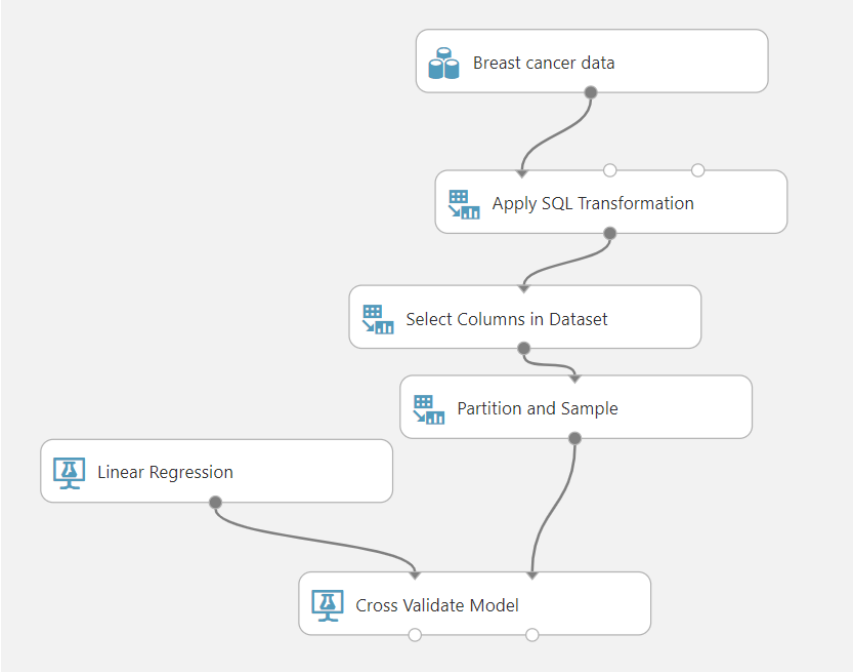
3.5 Microsoft Azure

Cloud v podaní Microsoftu s názvom Azure poskytuje viac ako 100 služieb, ktoré sú v čase písania tejto práce rozdelené do 18 produktových oblastí. Napríklad analytické služby, dátové úložiská, virtuálne siete, vývoj, bezpečnosť, databázy, manažment atď. Podobne ako aj na Google Cloude, tak aj tu nájdeme aplikácie umelej inteligencie a strojového učenia v nástrojoch pre strojový preklad, rozpoznávania reči a objektov. Pre tvorbu strojovo učiacich modelov sme použili Microsof Azure Machine Learning Studio. [21]

3.5.1 Microsoft Azure Machine Learning Studio

Microsoft Azure Machine Learning Studio poskytuje komplexný balík strojového učenia a dátovej analýzy. Do značnej miery sa odlišuje od Googlu Datalabu, kde bolo nutné ručne písať scripty. V aplikácií od Microsoftu nemusíme písať žiaden kód ak nechceme a celé prostredie je interaktívne a založené na moduloch, kde každý modul predstavuje určitú funkciu, ktorú vykonáva. Aj neskúsený užívateľ môže vytvárať pokročilé modely strojového učenia vďaka priateľskému užívateľskému prostrediu. Ďalej je možné pridať si modul, ktorý vie vykonať buď R script alebo Python script.

Na obrázku 12 môžeme vidieť model lineárnej regresie v prostredí Azure Machine Learning Studio. Nie je potrebné písať žiaden kód a model sa skladá z moduloch, kde každý modul má nejakú funkcionalitu.



Obrázok 12: Lineárna regresia v Azure Machine Learning studio

4 Zber dát

Riadenie železničnej dopravy spracovanie elektronickej dopravnej dokumentácie na území Českej a Slovenskej republiky zabezpečuje informačný systém GTN. Ten nahrádza pôvodné písomné rozpisy príchodov a odchodov vlakov. Zobrazuje operátorovi dopravy aktuálny stav na trati, spolu s plánovaným pohybom sa jednotlivých vlakových súprav v budúcnosti. Interaktívne si operátor vie zobraziť náhľad na celkovú dopravu medzi viacerými železničnými uzlami. Alebo si priblížiť a zobraziť podrobnosti o stanici, zastávke, odbočke, trati, vlakovej súprave, atď... V prípade stanice sa zobrazia podrobné informácie o počte koľají a prepojeniami medzi nimi. Počas prevádzky je dôležité udržiavať celkovú dopravu v pohybe a podľa plánu, prípadne sa prispôsobiť aktuálnym meškaniam a zároveň riadiť dopravu detailne v jednotlivých staniciach. Určovať presnú vlakovú cestu a miesto zastavenia v stanici.

Keďže človek nie je neomylný, tak ako všade aj tu sa stávajú nehody. Preto informačný systém GTN obsahuje modul ELDODO, ktorý zaznamenáva nie len históriu pohybu vlakov, ale zároveň zaznamenáva všetky pokyny a celé riadenie dopravy. Vďaka tomu sa dajú rekonštruovať a zistiť chyby, ktoré nastali a viedli k mimoriadnostiam. Pre potreby nášho modelu sú potrebné informácie o prejazdoch a presných časoch, kedy vlak prišiel a odišiel zo stanice. Modul ELDODO zapisuje všetky získané informácie do Oracle databázy. Tie sme si následne stiahli a extrahovali pre nás dôležité informácie. Tie sme transformovali do podoby záznamov vhodných pre ďalšie spracovanie. Každý záznam obsahoval kompletné údaje o jazde vlaku v medzistaničnom úseku a pozostával z týchto údajov:

- FromId – Id stanice z ktorej vlak vychádza
- FromName – Meno stanice z ktorej vlak vychádza
- ToId – Id cieľovej stanice
- ToName – Meno cieľovej stanice
- TrainId – Identifikácia vlaku
- TrainNumber – Číslo vlakovej súpravy
- TrainType – Typ vlakovej súpravy
- Weight – Váha celej vlakovej súpravy [t]
- Length – Dĺžka celej vlakovej súpravy [m]
- CarCount – Počet vagónov vlakovej súpravy

- AxisCount – Počet náprav vlakovej súpravy
- EngineType – Typ lokomotívy
- SectIdx – Číslo úseku na plánovanej trate jazdnej súpravy
- DepRealTime – Skutočný čas odchodu z východzej stanice
- DepILS – Zdroj skutočného odjazdu je z dôveryhodného zdroja
- ArrRealTime – Skutočný čas príchodu do cieľovej stanice
- ArrILS – Zdroj skutočného príjazdu je z dôveryhodného zdroja
- DepPlanTime – Plánovaný čas odchodu vlaku
- ArrPlanTime – Plánovaný čas príchodu vlaku
- LengthSect – Dĺžka železničného úseku [km]
- DriverId – Identifikácia rušňovodiča

V práci sú použité dáta z traťového úseku s celkovým počtom 22 staníc. Ich zoznam s identifikačným číslom a počtom záznamov je zobrazený v tabuľke 1 zoradený podľa počtu výskytov v dátach.

POČET	ID	MENO
499 094	34694000	Studenka
485 560	33722000	Hranice na Mor.
462 341	34652800	Prosenice
461 803	34804500	Suchdol nad Odr.
459 137	34042200	Lipnik nad Becv.
458 108	33472200	Drahotuse
451 328	38314100	Polanka n. O.
445 089	33654500	Jistebnik
394 366	34544700	Polom
287 524	34434100	Ostrava-Svinov
179 748	38274700	Odb.Odra
142 706	34662700	Prerov os.n.
116 138	38062600	Odb Skalka
115 387	38042800	Vyh Dluhonice
92 175	35044700	Ostrava-Vitkovice
58 054	33732900	Hranice mesto
46 836	34601500	Sedl.-Bartosovice
24 699	34144600	Novy Jicin mesto
24 360	33064700	Bilovec
22 034	34164400	Odry
17 216	33404500	Fulnek
6 342	34174300	Hermanky
5 474	35004100	Vitkov
4 898	33094400	Svatonovice
2 428	33084500	Budisov n.Budis.

1 651	33722002	Cement Hranice

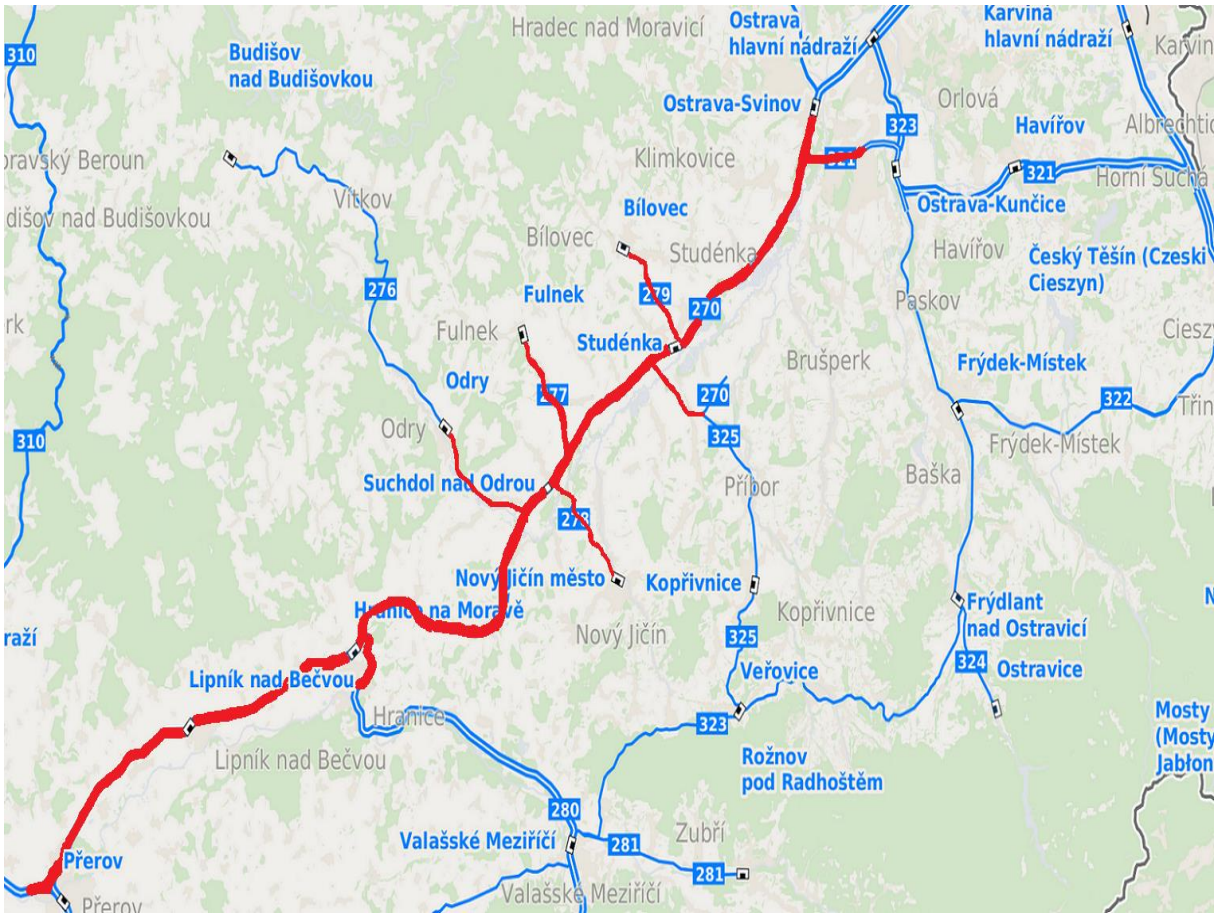
Tabuľka 1: Zoznam staníc

Za časové obdobie 5 rokov. Sme nazbierali 5 264 496 záznamov. Dáta sú v rozmedzí od začiatku roka 2016 do konca januára 2021. Počas tohto obdobia premávalo niekoľko druhov vlakov. Ich zoznam, početnosť a opis je zobrazený v tabuľke 2.

POČET	DRUH	POPIS
1 409 521	Ex	Expresný osobný
1 076 633	Os	Osobný
1 038 331	Pn	Prebežný nákladný
832 989	Nex	Nákladný expres
476 455	R	Rýchlik osobný
231 100	Lv	Lokomotivny vlak
59 979	Sluz	Služobný
55 764	Sv	Súpravový vlak
40 583	Sp	Zrýchlený osobný
32 222	Mn	Manipulačný
9 587	Vlec	Vlečkový nákladný
1 332	PMD	Posuny medzi dopravcami

Tabuľka 2: Druhy vlakov

Mapu železničných tratí spolu s vyznačenými úsekmi, z ktorých sme stiahli dáta sú vyznačené na obrázku 13. Jedná sa o Olomoucký a Moravsko-sliezsky kraj Českej republiky. V dátach je zahrnutá hlavná trať medzi mestami Přerov a Ostrava ale aj pripojené lokálne trate, zabezpečujúce menej frekventovanú premávku do okolitých miest a dedín.



Obrázok 13: Mapa železničných tratí

5 Analýza vstupných dát

Medzi dôležitejšie časti v procese vytvárania predikčného modelu patrí analýza dát. Na analýzu dát sa používajú rozličné štatistické metódy, nezanedbateľné je však aj použitie „sensus communis“ (zdravého rozumu).

Pri pohľade na vlastnosti vlakov, vieme pomocou plánovaných a skutočných odchodov a príchodov vlakov dopočítať ďalšie skryté informácie, medzi ktoré patrí:

- Meškanie pri príchode, odchode
- Predpokladaná jazdná doba vlaku
- Skutočná jazdná doba vlaku
- Zmena meškania na trati
- Priemerná plánovaná a skutočná rýchlosť na úseku

Pomocou spájania záznamov pri využití rovnakého identifikátora vlaku a zhodných odchádzajúcich a prichádzajúcich bodov vieme rekonštruovať celú jazdu konkrétneho vlaku v našej riadenej železničnej oblasti.

5.1 Analýza dát využitím jazyka Python

Python je vysoko úrovňový interpretovaný programovací jazyk s otvoreným zdrojovým kódom poskytujúci výborný prístup k objektovo orientovanému programovaniu. Aj keď jazyk nebol pôvodne špecificky navrhnutý pre dátovú analýzu, za posledných pár rokov sa podľa Jakea VanderPlasa [22] z neho stal prvotriedny nástroj na riešenie problémov v oblasti dátovej vedy. Hlavným dôvodom prečo bol Python tak rýchlo a široko prijatý v komunite vedy a rozvoja je relatívne ľahké používanie a jednoduchá syntax vyhovujúca osobám bez rozsiahlej inžinierskej praxe. Užitočnosť jazyka primárne spočíva v použití rozšíriteľných balíčkov, ktoré poskytujú nástroje pre analýzu a vizualizáciu veľkých súborov dát.

Medzi najviac používané balíčky jazyka Python pre analýzu dát patria Numpy, Pandas a Matplotlib [22].

Numpy

NumPy (numerical python) je knižnica, ktorá poskytuje matematické funkcie určené na spracovanie vysoko rozmerných polí. Obsahuje mnoho metód pre prácu s poliami a nástrojov lineárnej algebry. Knižnica uľahčuje prácu s vysoko rozmernými poľami a maticami.

Pandas

Pandas je jednou z najpopulárnejších knižníc pre manipuláciu a analýzu. Poskytuje mnoho užitočných nástrojov na manipuláciu veľkého počtu štruktúrovaných dát. Knižnica je relatívne nová a rozširuje funkcionalitu NumPy. Umožňuje využiť efektívnu implenentáciu dátovej štruktúry *DataFrame*. *DataFrame* je v podstate viacrozmerné pole rozšírené o označenie stĺpcov a riadkov. Pandas implementuje užitočné operácie na dátach, ktoré môžu byť povedomé používateľom databázových systémov a tabuľkových programov. Príkladom takýchto operácií sú napr.:

- Zlúčenie a spájanie súborov dát.
- Objekt *DataFrame* pre manipuláciu dát s integrovaným indexovaním.
- Filtrácia dát.
- Čítanie a zapisovanie dát v rôznych dátových štruktúrach a formátoch súborov.

Matplotlib

Matplotlib je knižnica, ktorú v našom prípade využijeme na vizualizáciu dát. Používa sa hlavne na vykresľovanie 2D grafov rôzneho druhu. Knižnica umožňuje modifikovať každý aspekt vygenerovaného diagramu, pohybovať sa na grafe a uložiť graf v požadovanom formáte.

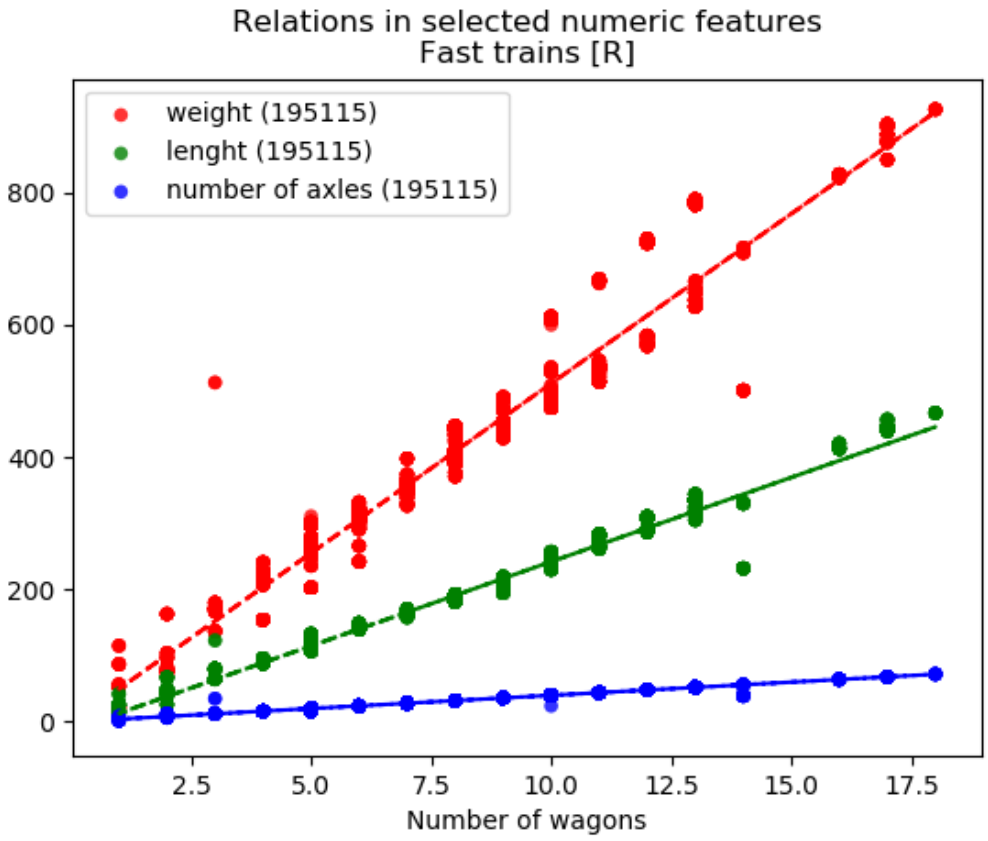
5.2 Vzťahy medzi vlastnosťami vlakov

Pri pohľade na zoznam vlastností vlakov v databáze s ktorými pracujeme, môžeme očakávať lineárnu závislosť medzi:

- Počet vagónov

- Počet náprav
- Dĺžka
- Váha

Po detailnejšom pohľade na vlastnosti, nedokážeme jednoznačne určiť vzťah medzi nimi. Vo väčšine prípadov sú zoradené v postupnosti od najnižšieho čísla. Počet vozňov, nasleduje počet náprav, ktorý je po väčšinou jeho štvornásobkom. Ďalším je dĺžka, ktorá sa líši od použitého typu vagóna a pohybuje sa okolo 50 násobku počtu vagónov. Posledná je váha, rovnako ako dĺžka je závislá na druhu vagónu. Avšak pohybuje sa okolo 55 násobku počtu vozňov. Detailnejšie to môžeme vidieť na grafe závislostí zobrazenom na obrázku 14. Kde v prípade počtu náprav sa hodnoty výrazne nevychyľujú od trendu. Vychýlené body môžu znamenať pripočítané nápravy lokomotívy. V prípade dĺžky a váhy sa jednoznačne trend čiary nedá určiť práve pre závislosti na použitých vagónoch o ktorých informácie nemáme.



Obrázok 14: Vzťah vybraných vlastností vlaku

5.3 Chybné vlastnosti

Učenie na základe nepresných alebo chybných informáciách nemôže viesť k správnemu výsledku. Preto tento krok patrí k veľmi dôležitým pri príprave súboru dát pre učenie a testovanie modelov umelej inteligencie.

Primárnym zdrojom dát pre náš model boli dáta z informačného a riadiaceho systému GTN, ktorý zbiera, koncentruje, zaznamenáva a vyhodnocuje dáta z rôznych ďalších externých systémov pre plánovanie a operatívne riadenie železničnej dopravy. Viaceré z týchto externých systémov (napr. systém pre operatívne riadenie ISOŘ, alebo dopravný denník) vykazujú chybovosť v ich dátach spôsobené ľudským faktorom, nakoľko vkladanie dát do týchto systémov je často realizované ručnou obsluhou a nie systémovým zberom údajov zo snímačov. S týmito chybami je treba pri analýze a čistení vstupných dát počítať a eliminovať ich dopady na učiaci sa model.

Systém GTN sám neopravuje chybné dáta prichádzajúce z externých systémov, ale ich zaznamenáva v podobe elektronickej dopravnej dokumentácie, na základe ktorej je možné spätne dohľadať pôvod vzniku chyby, čím následne pomáha správcovi železničnej infraštruktúry (v ČR Správa železnic, v SR Železnice slovenskej republiky) prijímať organizačno-technické opatrenia pre skvalitňovanie riadiacich procesov v železničnej doprave.

Aby sme získali čo najlepšie dáta pre náš model, pokúsili sme sa analyzovať chyby dát prichádzajúcich do GTN, kategorizovať ich do skupín a u jednotlivých skupín hľadať možné riešenia pre ich opravy, alebo pre minimalizáciu ich dopadu na učiaci sa model.

Na základe vzťahov medzi vlastnosťami vlakov v predchádzajúcej kapitole, môžeme definovať rozmedzie akceptovateľných hodnôt pre daný počet vozňov. Keďže táto hodnota je zadávaná ručne, zodpovedá za ňu obsluha a môže sa stať že v rýchlosti zabudne pridať desatinu čiarku, alebo omylom prehodí hodnoty pri rýchlom zadávaní (napr. namiesto dĺžky vlaku je uvedená hodnota počtu náprav a podobne) čo sú bežné a typické chyby spôsobené ľudským faktorom. V prípade, že by počtu vozňov nekorešpondovala váha, dĺžka ani počet náprav, využitím redundancie a Hamingovho kódu, vieme pri 4 hodnotách jednu opraviť a dve identifikovať.

Východzia a cieľová stanica sú kategorické premenné. V tomto prípade sa nedá vytvoriť vzorec, ktorý by dokázal rozpoznať nesprávne zadanú hodnotu prípadne by

určoval rozsah akceptovateľných hodnôt. V tomto prípade je lepšie použitie grafu železničnej trate. Vďaka železničnej sieti vieme definovať susedné stanice a následne povoliť záznamy len medzi susednými stanicami. Po vytvorení grafu s počtom vrcholov 22 vidíme, že obsahuje 23 hrán, v našom prípade je hrana definovaná ako koľaj medzi stanicami. Všetky trate podporujú premávku v oboch smeroch to nám dáva dokopy 46 kombinácii vychádzajúceho a koncového bodu pre vlakový záznam.

5.3.1 Konkrétne chyby odhalené pri zadávaní parametrov vlaku

Chýbajúca vlastnosť vlaku počas jazdy

Konkrétny vlak s unikátnym číslom má zadané všetky vlastnosti, avšak sa stáva že počas jazdy neskorších staniciach tieto vlastnosti zmizli, respektíve sú zadané ako nulové neplatné hodnoty, nakoľko do GTN prišli v nespracovateľnom stave (napr. namiesto očakávanej číselnej hodnoty prišiel text).

Duplicitné záznamy

Tieto záznamy je potrebné odstrániť pred vytvorením primárneho kľúča. V databáze sa nachádzajú záznamy s rovnakým číslom vlaku, času odchodu a tým aj ostatných hodnôt. Keďže ako unikátnu kombináciu v databáze používame číslo vlaku a skutočný čas odchodu, je potrebné tieto záznamy redukovať a ponechať len jeden reprezentatívny.

Podozrivo presné jazdy vlakov

V prípade, že železničná stanica nemala elektronický zabezpečovací systém, ktorý presne zaznamenáva prechod vlaku stanicou a jeho pobyty na koľajach. Prípadne bol dočasne odstavený, je možné zaznamenať príchod vlaku aj ručne tzv. manuálnou obsluhou. V tomto prípade je najjednoduchšie skopírovať plánovanú jazdnú dobu a zadať ju aj ako skutočnú. Síce ide o veľmi malé percento takéhoto výskytu, ale pre komplexnosť sme analyzovali aj tieto situácie.

Pri tejto analýze nie je podstatné, ktorú metriku chybovosti zvolíme, pretože v prípade nulovej chyby sa nelíšia ich hodnoty. Dôležitosť zohráva v prípade chybných dát, kde každá metrika je citlivá na iný druh chybovosti. Pre ukážku sme zobrazili v tabuľke 3 počet záznamov na danej trati. Percentuálne rozdelenie vlakov, ktorých skutočná jazdná doba bola rovnaká ako plánovaná. Na základe počtu vieme podokladať,

že to nebude náhoda a takýchto záznamov sa nachádza viacej. Posledný stĺpec zobrazuje priemernú jazdnú dobu medzi danými stanicami a tým reflektuje aj ich vzdialenosť.

Z - Do stanice	Počet vlakov	percent vlakov kde skutočná jazdná doba je rovná teoretickej [%]	Jazdná doba [s]
Prosenice - Výh Dluhonice	90366	0.998	422
Odb.Odra - Ostrava-Svinov	57411	0.996	226
Pøerov os.n. - Výh Dluhonice	16132	0.995	227
Polanka n. O. - Ostrava-Svinov	167693	0.994	166
Prosenice - Pøerov os.n.	112727	0.992	390
Odb.Odra - Ostrava-Vítkovice	70726	0.991	236
Odb Skalka - Hranice mìsto	43548	0.991	180
Výh Dluhonice - Pøerov os.n.	16425	0.981	212

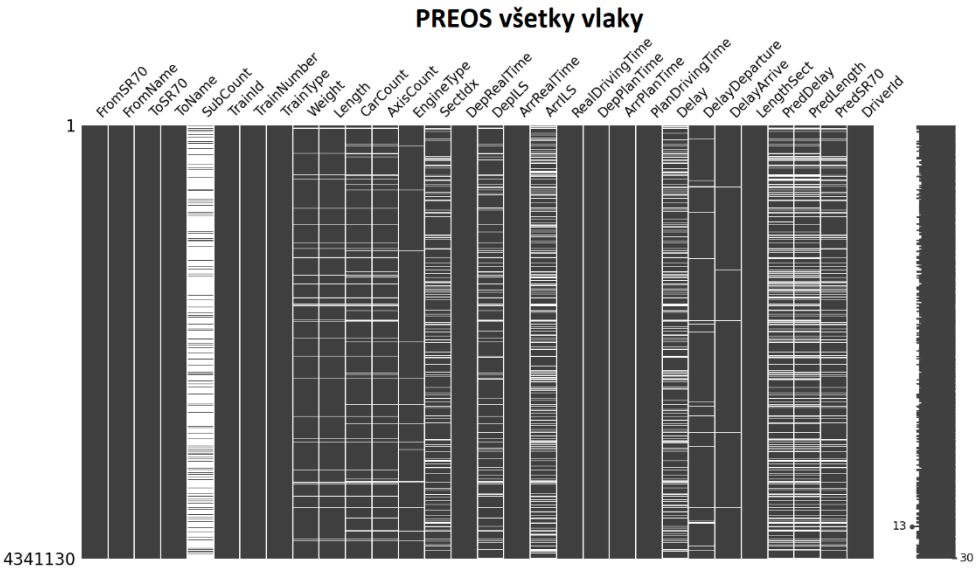
Tabuľka 3: Trasy s nulovou chybou

V prípade korelačnej matici by sa korelácia medzi teoretickou a skutočnou jazdnou dobou blížila hodnote 1. V takomto prípade nie je potrebné používať umelú inteligenciu, ktorá by riešenie úlohy zbytočne komplikovala, ale postačí algoritmus, ktorý v prípade detekcie jednej z trás z tabuľky 3 použije teoretickú jazdnú dobu.

5.4 Chýbajúce vlastnosti

Nezadaná hodnota, prázdny reťazec alebo reťazec obsahujúci len biele znaky ako medzera, tabulátor alebo nový riadok považujeme za chýbajúci údaj. Podľa obrázka 15 vidíme, že najčastejšie chýbajúcou hodnotou je telefónne číslo stroj vedúceho, ktoré sme využili pre identifikáciu rušňovodiča. Na druhú stranu vlastnosti, ktoré sú všetky zadané patrí vychádzajúci a koncový bod, id, číslo a druh vlaku, plánovaný čas odchodu a príchodu. Veľké percento zadaných hodnôt je aj v prípade skutočného odchodu a príchodu. Keďže tieto vlastnosti sú pre nás nevyhnutnou súčasťou a bez nich nedokážeme vypočítať jazdnú dobu, budú záznamy s nezadanou skutočnou jazdnou

dobou vymazané a dosiahnuté 100% pokrytie skutočného príchodu a odchodu z/do stanice.



Obrázok 15: Chýbajúce hodnoty

5.5 Korelácie a asociácie medzi premennými

Súbor dát sme vyfiltrovali, aby sa v ňom nachádzali iba vlaky osobnej dopravy, pretože tie obsahujú najmenej chybných dát. Následne sme odstránili všetky záznamy v ktorých absentovali údaje ktorejkoľvek premennej.

V našom prípade sa snažíme hľadať lineárny vzťah medzi numerickými premennými, takže využijeme poznatky z odseku 2.6.1. Pre numerické premenné sme pomocou funkcie knižnice pandas v jazyku python vypočítali korelačnú maticu, ktorú sme zapísali do tabuľky a silné vzťahy medzi premennými farebne zvýraznili. Tieto korelácie je možné vidieť v tabuľke 4.

	Hmot	Dlzka	PN	PV	Gvd	CasC	MO	MP
Hmot		0,99	0,97	0,97	-0,31	-0,29	0,04	0,04
Dlzka	0,99		0,97	0,97	-0,29	-0,27	0,04	0,04
PN	0,97	0,97		0,96	-0,28	-0,27	0,03	0,03
PV	0,97	0,97	0,96		-0,22	-0,21	0,03	0,03
Gvd	-0,31	-0,29	-0,28	-0,22		0,89	-0,02	-0,02

CasC	-0,29	-0,27	-0,27	-0,21	0,89		-0,02	-0,01
MO	0,04	0,04	0,03	0,03	-0,02	-0,02		1,00
MP	0,04	0,04	0,03	0,03	-0,02	-0,01	1,00	

Tabuľka 4: Korelačná matica

Skratky:

PN – počet náprav

PV – počet vozňov

GVD – teoretická jazdná doba

CasC – reálna jazdná doba

MO – meškanie vlaku pri odchode

MP – meškanie vlaku pri príchode

Najsilnejší vzťah so závislou premennou má premenná, ktorá reprezentuje hmotnosť vlaku. Ďalej sme sa dozvedeli, že táto premenná má silnú závislosť s premennými dĺžka vlaku, počet náprav vlaku, počet vozňov vlaku. Táto lineárna závislosť mala hodnotu Pearsonovho koeficientu nad 0,95.

Pre kategorické premenné sme vypočítali silu asociácie pomocou Cramérovho V asociačného koeficientu. Výsledné hodnoty sme zapísali do matice, ktorá je zobrazená v tabuľke 5. Veľmi silnú asociáciu s ostatnými kategorickými premennými má premenná „Druh“. Z toho vyplýva že rušňovodiči aj lokomotívy sú špecifické pre jednotlivé druhy vlakov a výrazne sa nevymieňajú

	Strojved.	**Loko**	**Druh**
Strojved.		0,31	0,90
Loko	0,31		0,99
Druh	0,90	0,99	

Tabuľka 5: Asociačná matica

Korelačný koeficient môže nadobúdať hodnoty <-1,1> a Cramérov asociačný koeficient <0,1>. Keďže neexistuje negatívna Cramérova asociácia, tak sme ponechali podobnú farebnú škálu na porovnanie sily asociácie, ako pri kladnej korelácií. Typy korelácií môžeme vidieť na obrázku 16.

1,00	Perfektná pozitívna korelácia
0,70	Silná pozitívna korelácia
0,50	Mierna pozitívna korelácia
0,30	Slabá pozitívna korelácia
0,00	Žiadna korelácia
-0,30	Slabá negatívna korelácia
-0,50	Mierna negatívna korelácia
-0,70	Silná negatívna korelácia
-1,00	Perfektná negatívna korelácia

Obrázok 16: Typy korelácií

Silne korelované alebo asociované nezávislé premenné tvoria fenomén, ktorý sa nazýva multikolinearita. To znamená, že v prípade zavedenia týchto premenných do modelu, sa môže výstup modelu drasticky meniť aj pri malej zmene modelu alebo dát. Takisto to spôsobuje pretrénovanie, čo znamená, že model bude produkovať správne výsledky, ale iba na dátach, na ktorých bol trénovaný. Nebude schopný variability. Tomuto fenoménu sa snažíme vyhnúť tým, že odstránime dané premenné z modelu. [11]

Pomocou tejto analýzy sme dospeli k záverom, že z modelu vylúčime premenné „Druh“, „Dlzka“, „PocNaprav“ a „PocVoznov“.

Z pozorovania sme zistili, že v prípade chýbajúcich hodnôt premennej „Hmot“ vieme dopočítať a skontrolovat tieto hodnoty pomocou iných premenných, s ktorými má silnú koreláciu.

5.6 Rozdelenie súboru dát na trénovanie a testovacie

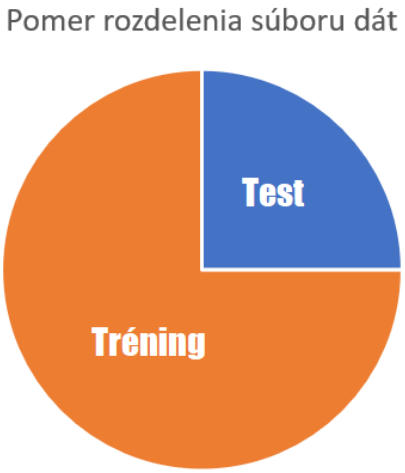
Na tréning predikčného modelu potrebujeme vzorku dát, na ktorej sa bude trénovať, a vzorku dát, na ktorej sa bude následne testovať a overovať úspešnosť predikcie. V praxi je zaužívané testovať model na inej množine dát, ako na ktorej bol trénovaný. Dôvodom je overenie toho, či je model správny. Testovanie modelu na tréningovej množine dát môže spôsobiť, že sa model bude javiť ako validný, ale v praxi bude pre neschopnosť variability nepoužiteľný. Tento jav sa nazýva pretrénovanosť. Teda model je pretrénovaný, ak nedokáže produkovať akceptovateľne presné predikcie na inej ako tréningovej množine dát.

Existuje niekoľko zaužívaných postupov delenia súboru dát na množinu dát pre tréning a test. V práci sme použili dva spôsoby delenia súboru dát.

5.6.1 Náhodné rozdelenie súboru

Náhodné rozdelenie súboru dát spočíva v delení súboru dát na dve časti na základe priradenej náhodnej hodnoty, podľa ktorej sa súbor rozdelí na dve časti.

Ako môžeme vidieť na obrázku 17, súbor dát je rozdelený na dve časti: 75% záznamov v tréningovej sade a 25% záznamov v testovacej sade.



Obrázok 17: Náhodné rozdelenie súboru dát

5.6.2 Krížová validácia

Krížová validácia je jedna z najpoužívanejších štatistických metód validácie modelov strojového učenia. Princíp spočíva v náhodnom rozdelení súboru dát na počet skupín približne rovnakej veľkosti, kde pre každú skupinu je vykonané testovanie modelu, ktorý bol trénovaný na zvyšných skupinách. Každá skupina bola raz použitá ako testovacia množina a „k-1“ krát použitá ako trénovacia množina, kde „k“ značí počet skupín. Výsledky zo všetkých testovaní „k“ modelov sa vyhodnotia spoločne.

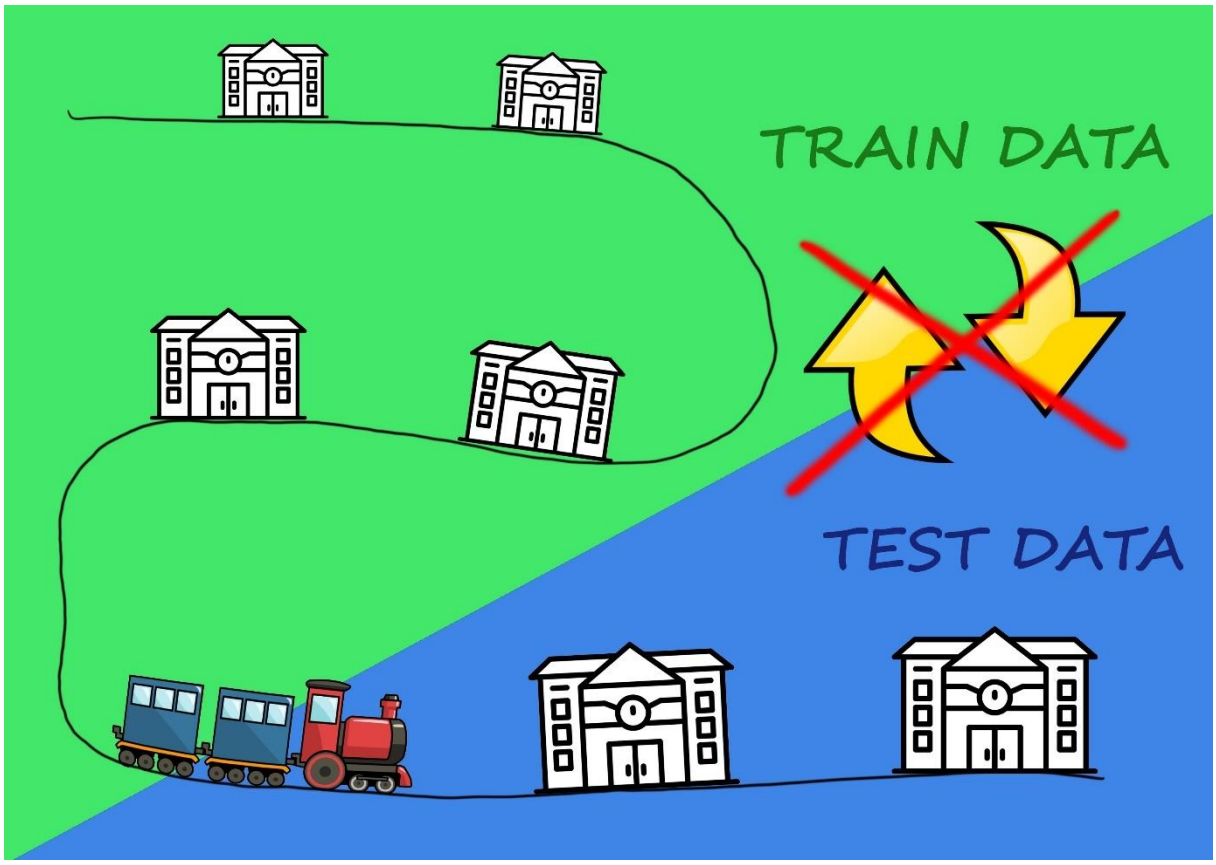
Ako je zobrazené v tabuľke 6, za hodnotu „k“ sme zvolili 4. Súbor dát sme rozdelili do štyroch skupín a pre každú skupinu sme vykonali testovanie modelu natrénovaného na zvyšných skupinách, vykonali sme teda tréning a test štyroch modelov.

MODEL	TEST	TRAIN	TRAIN	TRAIN
	TRAIN	TEST	TRAIN	TRAIN
	TRAIN	TRAIN	TEST	TRAIN
	TRAIN	TRAIN	TRAIN	TEST

Tabuľka 6: Rozdelenie súboru dát krížovou validáciou

5.6.3 Problém náhodného rozdelenia

Dáta obsahujú informácie o vlakoch a ich odchodoch. To znamená že existujú prepojenia medzi jednotlivými záznamami. V prípade že porušíme tieto vlastnosti, môže sa stať že model ohodnotíme ako veľmi dobrý. Avšak odhady budú ovplyvnené tým, že počas trénovania sa model dozvedel o budúcnosti daného vlaku. Napríklad ak vlak počas svojej cesty prechádza niekoľko zastávok. Nemôže sa stať, že niektoré úseky vlaku budú v testovacej množine a súčasne nasledujúce úseky, ktoré nepriamo obsahujú informácie o predchádzajúcich úsekoch budú v trénovanej skupine. V takom prípade sa môže stať že model je oboznámený, že na konci jazdy vlak nebude meškať. Vďaka tejto informácii vieme s dosť veľkou presnosťou povedať, že vlak nemešká ani v polovici jazdy. Ale tento prípad nie je korektný stav. Je to ako keby sme predpovedali súčasnosť s tým že poznáme budúcnosť. Preto treba dávať pozor na vzťahy v daných skupinách. Jedným z riešení je rozdeliť súbor dát na základe id vlaku. To zabezpečí, že celá cesta vlaku sa dostane do rovnakej skupiny a predíde sa situácii opísanej pred chvíľou. Podobnej situácii sa nemôžeme vyhnúť v prípade krížovej validácie kedy sa aj najstarší záznam musí v jednom kroku testovania dostať do testovacej množiny a porušiť časovanie. Avšak v prípade obsiahnutia všetkých záznamov o danom vlaku v testovacej skupine, nevzniká až taký veľký problém. Pretože informácia, že zajtra bude meškať vlak má minimálny súvis s aktuálnymi odjazdami vlakov.



Obrázok 18: Problém náhodného rozdelenia dát

6 Filtrovanie a oprava chybných záznamov

Kvalita strojového učenia je obmedzená kvalitou vstupných dát. V prípade, že vstupné dáta nie sú dostatočne správne a obsiahle. Nemôžeme očakávať dobré výsledky. Samozrejme vstupné dáta nie sú zárukou kvality. Aj ostatné kroky počas vývoja sú dôležité, ale stavať na zlých základoch už smeruje k neúspechu. V kapitole 5 Analýza vstupných dát sme opísali spôsoby, ako vyhľadať chybné záznamy. A v tejto kapitole sa sústredíme na spôsoby, ako vstupné dáta opraviť a zlepšiť.

Medián

Skupinu záznamov s vyplnenou vlastnosťou, ktorú chceme dopočítať zoradíme podľa veľkosti a hodnota v strede tohto radu sa nazýva medián. Je to konkrétne číslo jedného zo záznamov, ktoré sa doplní na všetky chýbajúce miesta danej vlastnosti.

Priemer

Podobne s vyfiltrovaných záznamov spracujeme požadovanú vlastnosť, avšak tento krát využijeme funkciu priemeru. Výslednú hodnotu doplníme do záznamov, ktoré danú vlastnosť nemajú zadanú. Problém riešenia je náchylnosť na extrémy. Tie môžu nastať chybnými dátami, ktoré sme neodhalili počas analýzy. Alebo špeciálne situácie prepravy kedy sa prepravoval neštandardný tovar.

Modus

Je najčastejšie vyskytovaná hodnota. V prípade reálnych čísel je výpočet *modus*-u v princípe nemožný. Jedným s riešením pre reálne čísla je použitie intervalov, ale v praxi sa všeobecne *modus* používa len výnimočne.

Špeciálne

Je založené na hlbšej znalosti dát, kde sa v našom prípade môžeme sústrediť napríklad na druh vlaku a tým sprísniť selekciu, čím získame podobnejšie a viac pravdepodobné hodnoty tej chýbajúcej. V našom prípade ak by nám chýbala hodnota váhy vlaku, tak pridané filtrovanie podľa druhu vlaku by zlepšilo výslednú hodnotu, keďže nákladné vlaky sa pohybujú v iných váhových rozmedziach ako osobné vlaky. Taktiež je možné k výpočtu použiť matematický vzorec alebo iný postup výpočtu. Avšak to záleží na skrytých vlastnostiach medzi hodnotami, ktoré užívateľ môže poznať vďaka hlbším znalostiam daného problému.

Kombinácie

Skutočnosť je zvyčajne oveľa viac komplikovaná ako simulácie, kde zanedbávame veľké množstvo menších javov. V prípade, že náš problém má špecifické vlastnosti, je možné použiť kombinácie vyššie uvedených. Napríklad pri osobných vlakov použiť priemer a u nákladných medián či iné kombinácie.

7 Implementácia

V tejto kapitole sa budeme venovať implementácii filtrovania, selekcii dát a umelej inteligencii v niekoľkých programovacích jazykoch a použitím rozdielnych frameworkov. Keďže budeme používať rovnaké vstupné dáta, tak nám umožní porovnať kvalitu jednotlivých implementácii. V prípade použitia rozdielneho jazyka a implementácie algoritmu lineárnej regresie, výsledky by nemali byť rozdielne, keďže postup je rovnaký. Vybrali sme preto frameworky, ktoré obsahujú implementáciu rôznych heuristík a tým aj výsledky by mali byť rozdielne a zaujímavejšie. Osobitná kapitola bude venovaná cloudovým riešeniam. Kde sa sústredíme na spôsob implementácie a použitia. Je to nový spôsob prístupu k výpočtom, ktorý by mal viesť k zníženiu nákladov a lepšiemu využitiu zdrojov informačných technológii.

V tabuľke 7 je zobrazený počet a názov detekovanej chyby, ktoré budú vysvetlené neskôr v tejto kapitole. Podľa počtu výskytov môžeme určiť spôsob riešenia a dôležitosť, či je potrebné sa tomu venovať, alebo je to nepatrná chyba a záznamy vymazať. Počet sa vždy odvíja od vstupného dataset-u, čo znamená, že jeden záznam môže obsahovať viac chýb a tým započítaný vo viacerých riadkoch.

Počet	Percent	Názov chyby
21158	0,402	Nespĺňajú rovnosť váha dĺžka...
1674	0,032	Posun o jeden deň
504	0,010	Strata váhy počas jazdy
529	0,010	Strata dĺžky počas jazdy
682	0,013	Strata poctu naprav počas jazdy
958	0,018	Strata postu vagónov počas jazdy
2235	0,042	Nulová alebo záporná skutočná jazdná doba
1861	0,035	Nulová alebo záporná plánovaná jazdná doba
8736	0,166	Duplicitný záznam
7566	0,144	Chýba plánovaný príchod
15417	0,293	Chýba plánovaný odchod

Tabuľka 7 Počet výskytov a druh detekovanej chyby

7.1 Korekcia

7.1.1 Nezadané hodnoty

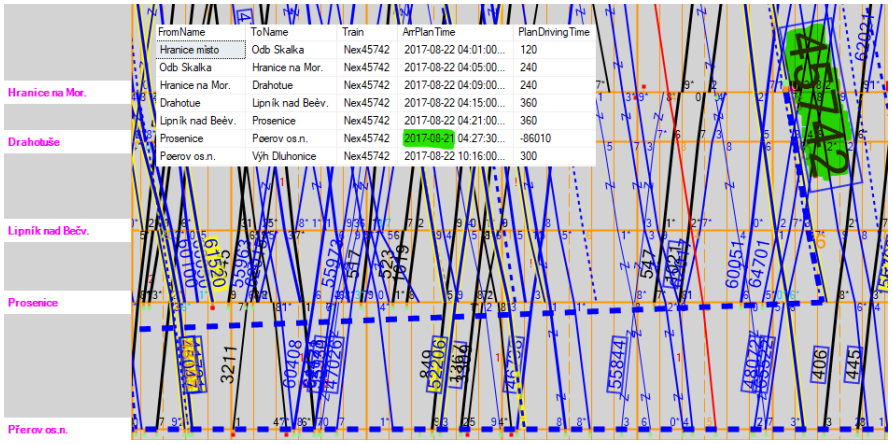
V programovaní sa za neznámu hodnotu používa dátový typ null pokiaľ je podporovaný. Pri importe z databázy sa táto hodnota v dátach nenachádza a preto sa

zvykne používať hodnota „0“ alebo „-1“. Prvým krokom pri úprave databáze zjednotíme túto informáciu a všetky výskyty zápornej hodnoty aktualizujeme na hodnotu „0“. V databáze sa nachádza 10 záznamov s hodnotou „-1“ a 334 000 s hodnotou „0“

```
UPDATE CZPREOS set CarCount = 0 where CarCount = -1
```

7.1.2 **Posun dátumu**

Všimli sme si niekoľko skutočných a plánovaných jazdných dôb v intervale +-80000 až 90000. Keďže databáza používa časové hodnoty v sekundách, 1 deň je reprezentovaný ako 86400 sekúnd. Pri detailnom pohľade na vlak časová stopa kopíruje predpokladanú hodnotu avšak dátum je posunutý. Príkladom je vlak Nex45742 na obrázku 19. Percentuálne k všetkým dátam je tento druh chyby veľmi ojedinelý. Najčastejšie sa objavuje v záznamoch s cielovou stanicou Přerov os.n. a stanici Studénka.



FromName	ToName	Train	ArrPlanTime	PlanDrivingTime
Hranice misto	Odb Skalka	Nex45742	2017-08-22 04:01:00...	120
Odb Skalka	Hranice na Mor.	Nex45742	2017-08-22 04:05:00...	240
Hranice na Mor.	Drahotue	Nex45742	2017-08-22 04:09:00...	240
Drahotue	Lipník nad Beèv.	Nex45742	2017-08-22 04:15:00...	360
Lipník nad Beèv.	Prosenice	Nex45742	2017-08-22 04:21:00...	360
Prosenice	Pøerov os.n.	Nex45742	2017-08-21 04:27:30...	-86010
Pøerov os.n.	Výh Dluhonice	Nex45742	2017-08-22 10:16:00...	300

Obrázok 19 Dátum záznamu posunutý o deň

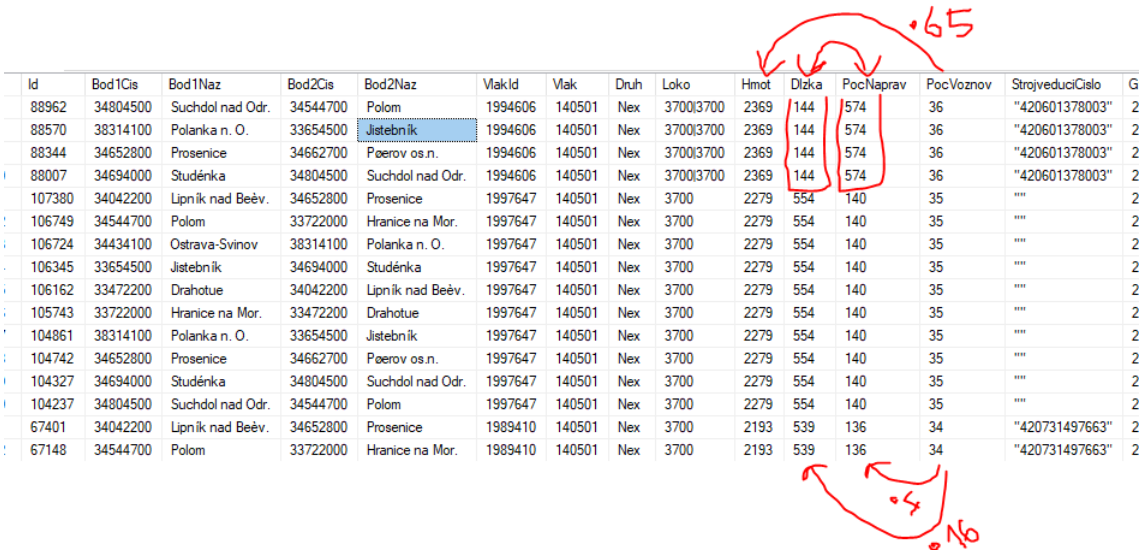
7.1.3 **Prehodené stĺpce v záznamoch**

Definícia vlakovej súpravy obsahuje štyri úzko súvisiace hodnoty. Dá sa očakávať, že ich zadávanie bude spojené v jednom mieste programu a preto sa v dátach nachádzajú tieto hodnoty prehodené. Ako príklad je zobrazený nákladný expres (Nex) číslo 140501 na obrázku 20, kde je zobrazená závislosť:

Počet voznov * 4 = počet náprav

Počet voznov * 16 = dĺžka

Počet voznov * 65 = hmotnosť



Id	Bod1Cis	Bod1Naz	Bod2Cis	Bod2Naz	VlakId	Vlak	Druh	Loko	Hmot	Dlzka	PocNaprav	PocVoznov	StrojveduciCislo	G
88962	34804500	Suchdol nad Odr.	34544700	Polom	1994606	140501	Nex	3700|3700	2369	144	574	36	"420601378003"	2
88570	38314100	Polanka n. O.	33654500	Jistebník	1994606	140501	Nex	3700|3700	2369	144	574	36	"420601378003"	2
88344	34652800	Prosenice	34662700	Pøerov os.n.	1994606	140501	Nex	3700|3700	2369	144	574	36	"420601378003"	2
88007	34694000	Studénka	34804500	Suchdol nad Odr.	1994606	140501	Nex	3700|3700	2369	144	574	36	"420601378003"	2
107380	34042200	Lipník nad Beèv.	34652800	Prosenice	1997647	140501	Nex	3700	2279	554	140	35	""	2
106749	34544700	Polom	33722000	Hranice na Mor.	1997647	140501	Nex	3700	2279	554	140	35	""	2
106724	34434100	Ostrava-Svinov	38314100	Polanka n. O.	1997647	140501	Nex	3700	2279	554	140	35	""	2
106345	33654500	Jistebník	34694000	Studénka	1997647	140501	Nex	3700	2279	554	140	35	""	2
106162	33472200	Drahotue	34042200	Lipník nad Beèv.	1997647	140501	Nex	3700	2279	554	140	35	""	2
105743	33722000	Hranice na Mor.	33472200	Drahotue	1997647	140501	Nex	3700	2279	554	140	35	""	2
104861	38314100	Polanka n. O.	33654500	Jistebník	1997647	140501	Nex	3700	2279	554	140	35	""	2
104742	34652800	Prosenice	34662700	Pøerov os.n.	1997647	140501	Nex	3700	2279	554	140	35	""	2
104327	34694000	Studénka	34804500	Suchdol nad Odr.	1997647	140501	Nex	3700	2279	554	140	35	""	2
104237	34804500	Suchdol nad Odr.	34544700	Polom	1997647	140501	Nex	3700	2279	554	140	35	""	2
67401	34042200	Lipník nad Beèv.	34652800	Prosenice	1989410	140501	Nex	3700	2193	539	136	34	"420731497663"	2
67148	34544700	Polom	33722000	Hranice na Mor.	1989410	140501	Nex	3700	2193	539	136	34	"420731497663"	2

Obrázok 20: Prehodené číselné stĺpce

Teda jeden vozeň danej vlakovej súpravy má 4 nápravy, dĺžku 16metrov a hmotnosť 65 ton a musí spĺňať závislosť počet voznov < počet náprav < dĺžka < hmotnosť. Počet záznamov, ktoré nespĺňajú rovnosť je 21158.

```
select * from CZPREOS where CarCount > AxisCount or AxisCount > Length or Length
> Weight
```

7.1.4 Vynechané hodnoty

V prípade osobného vlaku číslo Os23309 sa do dátumu 9. októbra 2018 neudávali parametre dĺžky, váhy, počtu náprav a vagónov ako je zobrazené na obrázku 21. Je zrejmé, keďže sa jedná o rovnaké vlaky v rovnakej stanici s rovnakým časom odchodu len iný deň že údaje neboli zadané alebo dostupné v systéme ISOŘ, odkiaľ prichádzajú informácie o parametroch vlaku, a pravdepodobne budú rovnaké pre všetky záznamy. Taktiež sa pravidlá a zodpovednosť za správne vyplnené údaje zvyšuje. Ako je na obrázku znázornené, je to z roku 2018 a novšie záznamy už podobné problémy neobsahujú.

Id	Bod1Cis	Bod1Naz	Bod2Cis	Bod2Naz	VlakId	Vlak	Druh	Loko	Hmot	Dlzka	PocNaprav	PocVoznov	StrojveduciCislo	GvdOdch	GvdPrich	Odch	Prich
217...	33064700	Bílovec	34694000	Studénka	2447983	23309	Os	8140	0	0	0	0	""	2018-10-05 07:28:00...	2018-10-05 07:39:00....	2018-10-05 07:30:00...	2018-10-05 07:41:00...
217...	33064700	Bílovec	34694000	Studénka	2448716	23309	Os	8140	0	0	0	0	""	2018-10-06 07:28:00...	2018-10-06 07:39:00....	2018-10-06 07:28:00...	2018-10-06 07:39:00...
217...	33064700	Bílovec	34694000	Studénka	2449414	23309	Os	8140	0	0	0	0	""	2018-10-07 07:28:00...	2018-10-07 07:39:00....	2018-10-07 07:30:00...	2018-10-07 07:41:00...
218...	33064700	Bílovec	34694000	Studénka	2450093	23309	Os	8140	0	0	0	0	""	2018-10-08 07:28:00...	2018-10-08 07:39:00....	2018-10-08 07:28:00...	2018-10-08 07:39:00...
218...	33064700	Bílovec	34694000	Studénka	2450871	23309	Os	8140	0	0	0	0	""	2018-10-09 07:28:00...	2018-10-09 07:39:00....	2018-10-09 07:26:00...	2018-10-09 07:37:00...
218...	33064700	Bílovec	34694000	Studénka	2451682	23309	Os	8140	47	29	4	2	"724162838"	2018-10-10 07:28:00...	2018-10-10 07:39:00....	2018-10-11 07:28:00...	2018-10-11 07:39:00...
219...	33064700	Bílovec	34694000	Studénka	2452535	23309	Os	8140	47	29	4	2	"725064148"	2018-10-11 07:28:00...	2018-10-11 07:39:00....	2018-10-11 07:28:00...	2018-10-11 07:39:00...
219...	33064700	Bílovec	34694000	Studénka	2453432	23309	Os	8140	47	29	4	2	"725064290"	2018-10-12 07:28:00...	2018-10-12 07:39:00....	2018-10-12 07:28:00...	2018-10-12 07:39:26...
219...	33064700	Bílovec	34694000	Studénka	2454311	23309	Os	8140	47	29	4	2	"724162838"	2018-10-13 07:28:00...	2018-10-13 07:39:00....	2018-10-13 07:28:00...	2018-10-13 07:40:06...
219...	33064700	Bílovec	34694000	Studénka	2455033	23309	Os	8140	47	29	4	2	"725064148"	2018-10-14 07:28:00...	2018-10-14 07:39:00....	2018-10-14 07:28:00...	2018-10-14 07:39:58...
220...	33064700	Bílovec	34694000	Studénka	2455804	23309	Os	8140	47	28	4	2	"725064148"	2018-10-15 07:28:00...	2018-10-15 07:39:00....	2018-10-15 07:28:00...	2018-10-15 07:40:27...

Obrázok 21: Chýbajúce dáta v pravidelnom vlaku

Zamerali sme sa na záznamy, v ktorých určitý druh a číslo vlaku majú v databáze záznamy s nulovou hodnotou aj zadanou jednou unikátnou hodnotou. V takom prípade sme existujúcu hodnotu doplnili do záznamov, kde chýbala. Použili sme SQL príkaz, ktorý zobrazí vlaky, obsahujúce váhu 0 (nezadaná hodnota)

```
select CONCAT(TrainType, TrainNumber) id, count(*) bcount from CZPREOS where
        Weight = 0 group by CONCAT(TrainType, TrainNumber)
```

a zároveň v databáze sa nachádza so zadanou jednou hodnotou.

```
select CONCAT(TrainType, TrainNumber), count(*) acount from CZPREOS where
Weight <> 0 group by CONCAT(TrainType, TrainNumber) having count(distinct weight)
= 1
```

následne po spojení týchto príkazov cez JOIN, sme získali zoznam vlakov, ktorým môžeme doplniť nezadanú váhu z iných záznamov rovnakého vlaku, kde daná váha bola zaznamenaná.

```
select aa.id, aa.acount, bb.bcount from
(select CONCAT(a.TrainType, ' ', a.TrainNumber) id, count(*) acount from CZPREOS
a where a.Weight <> 0 group by CONCAT(a.TrainType, ' ', a.TrainNumber) having
count(distinct a.weight) = 1) aa
Join
(select CONCAT(b.TrainType, ' ', b.TrainNumber) id, count(*) bcount from CZPREOS
b where b.Weight = 0 group by CONCAT(b.TrainType, ' ', b.TrainNumber)) bb
    on aa.id = bb.id
```

Týmto spôsobom sme získali zoznam vlakov, ktorým treba opraviť váhu a ich opčet je 504. Následnú korekciu sme spravili príkazom, ktorý vnútorne vyhľadá daný typ a číslo vlaku so správnou (nenulovou) váhou.

```
update CZPREOS set weight = (select top(1) weight from CZPREOS where Weight <>
0 and TrainType = 'Lv' and TrainNumber='57249') where TrainType = 'Lv' and
TrainNumber='57249'
```

Podobným spôsobom sme postupovali s dĺžkou vlaku (529 výskytov) a počtom náprav (682 výskytov) a vagónov (958 výskytov).

7.1.5 Zmeny jazdnej súpravy počas jazdy

Tento druh chyby je veľmi ťažko identifikovať. Jazdná súprava môže počas svojej cesty meniť počet vagónov a to hlavne pri nákladnej doprave. Napríklad vlak PN48056 počas svojej jazdy pribral druhú lokomotívu s označením 2230 a tým sa zvýšila aj jeho hmotnosť a dĺžka.

	Id	Bod1Cis	Bod1Naz	Bod2Cis	Bod2Naz	VlakId	Vlak	Druh	Loko	Hmot	Dlzka	PocNaprav	PocVoznov	StrojveduciCislo	GvdOdch
1	1251033	34662700	Pøerov os.n.	34652800	Prosenice	2245720	48056	Pn	3266	1708	519	100	25	"420"	2018-01-1
2	1251408	34652800	Prosenice	34042200	Lipník nad Beèv.	2245720	48056	Pn	2230|3266	1708	519	100	25	"420"	2018-01-1
3	1254255	34042200	Lipník nad Beèv.	33472200	Drahotue	2245720	48056	Pn	2230|3266	1708	519	100	25	"420"	2018-01-1
4	1253040	33472200	Drahotue	33722000	Hranice na Mor.	2245720	48056	Pn	2230|3266	1708	519	100	25	"420"	2018-01-1
5	1254036	33722000	Hranice na Mor.	34544700	Polom	2245720	48056	Pn	2230|3266	1708	519	100	25	"420"	2018-01-1
6	1253754	34544700	Polom	34804500	Suchdol nad Odr.	2245720	48056	Pn	2230|3266	1708	519	100	25	"420"	2018-01-1
7	1252240	34804500	Suchdol nad Odr.	34694000	Studénka	2245720	48056	Pn	2230|3266	1701	518	100	25	"420"	2018-01-1
8	1250944	34694000	Studénka	33654500	Jistebník	2245720	48056	Pn	3266	1701	518	100	25	"420"	2018-01-1
9	1252625	33654500	Jistebník	38314100	Polanka n. O.	2245720	48056	Pn	3266	1701	518	100	25	"420"	2018-01-1
10	1252054	38314100	Polanka n. O.	34434100	Ostrava-Svinov	2245720	48056	Pn	3266	1701	518	100	25	"420"	2018-01-1

Obrázok 22 Zmena jazdnej súpravy počas jazdy

Po hlbšom preskúmaní viacerých záznamov, sme našli vlaky, ktoré počas svojej jazdnej doby zmenili zostavu aj 4 krát. V týchto prípadoch sa s lineárnou závislosťou sa menil počet vagónov, dĺžka aj váha jazdnej súpravy. Aj po ručnom preskúmaní jazdných dôb, typu vlaku a záznamoch o nich sme zistili. Že je to štandardný postup, kde vlaková súprava má dlhší pobyt v stanici, potrebný na naloženie, preloženie alebo vyloženie nákladu. Preto aj situácia z obrázku 22 znázorňuje štandardnú prevádzku železničnej dopravy.

7.2 Filtrovanie

7.2.1 Vymazať záznamy medzi nesusednými stanicami

V prípade že sa názov výjazdovej a cieľovej stanice zhoduje, jedná sa o vlaky ktoré menia koľaj v rámci stanice. Vyšli na medzistaničný úsek a vrátia sa na inú koľaj do rovnakej stanice. Tieto záznamy nie sú pre nás zaujímavé.

V databáze sa nachádzajú aj záznamy medzi stanicami, ktoré nie sú vedľa seba. Takýchto záznamov sa nachádzalo 23 759, čo je len okolo pol percenta a zároveň sa v nich nachádza 10 rôznych kombinácii miesta odjazdu a príjazdu. Tým je ťažko natrénovať kvalitný model a odhadovať jazdné. Preto sa sústredíme na frekventovanejšie trate.

Stanice mimo hlavnú vetvu

V riadenej oblasti PREOS. Ktorá je medzi Přerovom a Ostravou sa nachádzajú tiež záznamy zo staníc, ktoré nie sú priamo spojené s hlavnou traťou, Konkrétne ide o stanice:

- 33084500 Budišov n.Budišovkou
- 33094400 Svatonovice
- 35004100 Vítkov

- 34174300 Heřmánky

Tým je hustota premávky v týchto staniciach výrazne nižšia. Najviac premávky je na hlavnej trati, kde premávajú všetky typy vlakov a je súčasťou aj medzinárodnej prepravy.

7.2.2 Neznáme druhy vlakov

Najviac zastúpené sú osobné a nákladné vlaky, ktoré sa ďalej delia na expresné, zrýchlené a iné. Avšak niektoré záznamy nemajú správne zadaný textový druh, ale obsahujú neznámy názov alebo číslo. Napríklad v oblasti Čadce, kde máme prístup k prejazdom vlakov aj počasiu, sa nachádzajú druhy vlakov ako: 32, 39, 40, 47, nesp. Ich celkový počet je len 72 záznamov z celkového počtu 341176. Reprezentuje to len 0,2% záznamov, čo mohlo vzniknúť testovaním trate alebo iným neštandardným prejazdom prípadne neštandardnou vlakovou súpravou.

7.2.3 Skutočná alebo plánovaná jazdná doba je menšia alebo rovná nule

Záporné alebo nulové jazdné doby by sa nemali v záznamoch objavovať. Záporná jazdná doba znamená, že vlak dorazil do cieľa skôr, ako vyštartoval. V prípade skutočnej jazdnej doby je to 2235 záznamov, v prípade plánovanej jazdnej doby 1861. Pravdepodobným zdrojom bude chyba synchronizácie času prejazdových zariadení, komunikačných protokolov alebo obsluhy, ktorá má možnosť zadať skutočný príchod alebo odchod aj ručne. Nepatrné množstvo záznamov malo zápornú jazdnú dobu v kombinácii s opačným poradím staníc, posunutím záznamu odjazdu o deň skôr ako sme už písali v kapitole 7.1.2 alebo spôsobené pravidelným posunom medzi letným a zimným časom. Takýchto záznamov sa v importovaných dátach nachádzalo 6130. Kde nesprávna plánovaná jazdná doba bola v 2625 prípadoch a skutočná jazdná doba v 4990 čo znamená, že 1485 záznamov malo chybnú aj skutočnú aj plánovanú jazdnú dobu. Aj v tomto prípade je percento výskytov uvedených prípadov nízke, avšak pre prípravu čo najlepších dát pre trénovanie modelu sme tieto záznamy eliminovali.

```
delete from CZPREOS where PlanDrivingTime <= 0 or RealDrivingTime <= 0
```

7.2.4 Duplikáte záznamy

Rozpoznať dva záznamy, ktoré reprezentujú rovnakú udalosť sa snažíme pomocou trainId, ktoré unikátne definuje vlakovú súpravu v celej databáze a času skutočného

odjazdu, keďže vlak nemôže v rovnaký čas mať dva záznamy o odchode. Odhalili sme 8736 duplicitných záznamov, kde maximálny počet rovnakých záznamov bol 11 ale vo väčšine prípadov sa nachádzali len 2 rovnaké.

```
group by CONCAT(TrainId, ' ', DepRealTime) having count(*) > 1
```

Aplikácia z ktorej sme sťahovali dáta (GTN) zbiera informácie zo zabezpečovacích zariadeniach snímajúcich prejazdy vlakov cez body na trati. Pomocou unikátneho identifikátora vlaku sú tieto záznamy o skutočných prejazdoch doplnené o parametre vlaku, ktoré sú dostupné z „Informačného systému operatívneho riadenia“ skratkou „ISOŘ“. K zlepšeniu predikcie a rozšíreniu vstupných dát sme využili prepojenie na IS Kango, ktoré obsahuje informácie o celoročnom grafikone a plánovaných odjazdoch a príjazdoch vlakov. Duplicitné záznamy mohli teda vzniknúť pri čítaní prejazdu vlaku, alebo počas odosielania dát zo zabezpečovacieho systému. Z bezpečnostných dôvodov sú všetky informácie zaznamenané v systéme GTN aj v prípade, že sa jedná o redundanciu.

7.2.5 Vymazať záznamy s chýbajúcimi dôležitými informáciami

Dôležité sú skutočné a plánované príchody a odchody vlakov. V prípade chýbajúceho plánu jazdy vlaku, sú to výnimočne stavy, kde vlak nemusí dodržiavať očakávané jazdné vlastnosti. Môže sa vyskytnúť aj chyba pri získavaní dát a plánovaná jazdná doba by sa dala doplniť v prípade že sa jedná o pravidelný spoj. Avšak týchto záznamov je len nepatrné množstvo. V prípade chýbajúcich časov skutočnej jazdy je nemožné určiť skutočnú jazdnú dobu a teda ďalej pracovať so záznamom. To vedie k vymazaniu daných dát. Skutočné príchody a odchody sú zadané vo všetkých záznamoch, počet nezadaných časov pre plánovaný odchod je 7566 a plánovaný príchod 15417

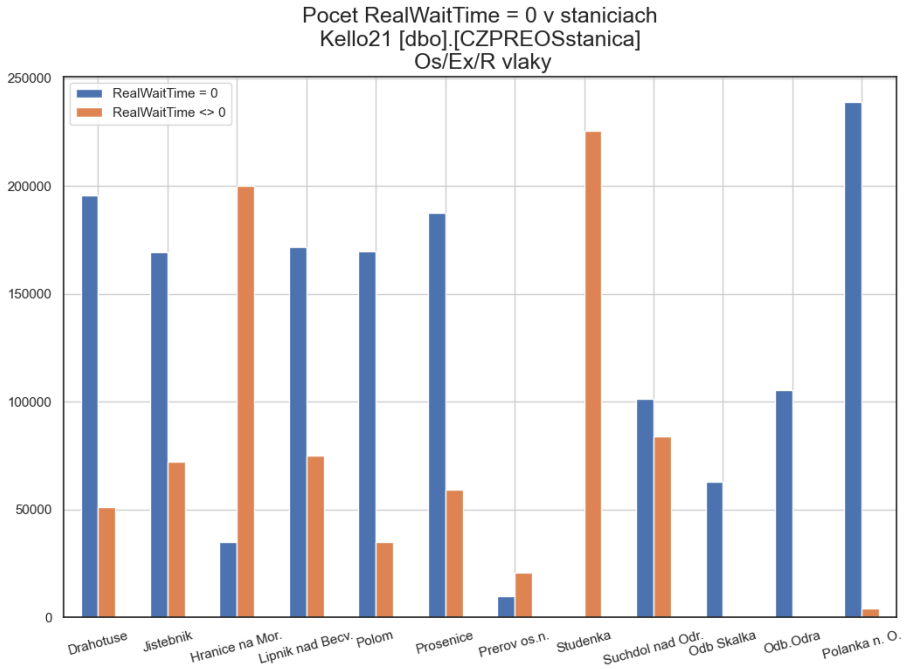
7.2.6 Vytvorenie unikátneho identifikátora

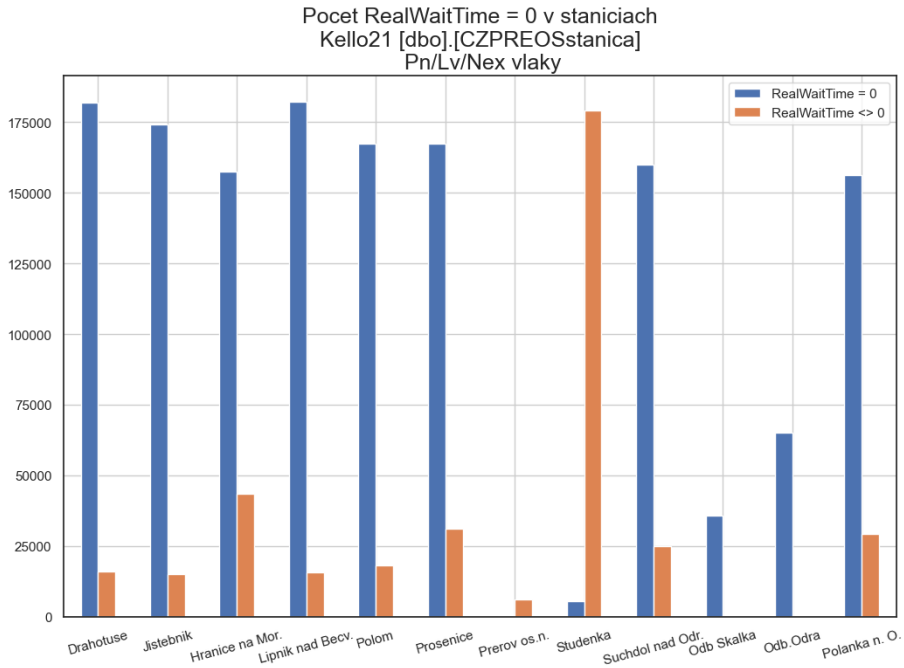
V dátach sa nachádza TrainId ktoré unikátne definuje vlakovú súpravu počas jej cesty. Ako prechádza stanicami, zaznamenáva sa niekoľko záznamov s rovnakou hodnotou TrainId, preto je potrebné vytvoriť unikátny identifikátor. Pri niektorých vlakoch sa stáva, že majú viac záznamov s rovnakým ID a vychádzajúcou stanicou (napr. kyvadlová doprava, návrat z trate, atď...).Teda môže sa vyskytnúť viac záznamov s rovnakým ID a vychádzajúcou stanicou. Pre model je však vhodné nekumulovať tieto opakujúce sa trasy do jedného vlaku, ale separovať ich do samostatných jázd vlaku. Preto

sme sa rozhodli vytvoriť unikátny identifikátor kombináciu ID a času skutočného odjazdu. Táto kombinácia musí byť unikátna a dva odchody toho istého vlaku v rovnaký čas nie sú možné.

7.2.7 **Prejazd stanicou**

Rozlišujeme vlaky, ktoré v stanici zastavujú a prechádzajú ňou. Sú to hlavne nákladné typy vlakov, ktorých trasa reprezentuje prepravu materiálu. Alebo sú to medzinárodné osobné vlaky, ktorých počet zastavení je redukovaný k zrýchleniu prepravy cestujúcich. Ak to premávka v týchto prípadoch dovoľuje, pokračuje vlak cez stanicu bez zastavenia. V databáze to reprezentujeme ako nulový pobyt v stanici, keďže čas príjazdu a odjazdu zo stanice je rovnaký. Týmto spôsobom vieme zobraziť početnosť vlakov pre každú stanicu, ktoré v nej zastavili, alebo ňou prešli. Na obrázku 23 sú znázornené pomery vlakov, ktoré stanicami prešli (modrá farba) a zastavili (oranžová). Horný graf znázorňuje nákladnú dopravu a osobná doprava je zobrazená pod ním.





*modrá – prejazd stanicou
oranžová – pobyt v stanici

Obrázok 23 Nákladné a osobné vlaky rozdelené na prejazd a pobyt v stanici

7.3 Spájanie záznamov

7.3.1 Viacerých riadených oblastí

Train ID je pre rovnaký vlak v inej oblasti rozdielny, preto je potrebné spárovať vlaky. Zvyčajne v oblasti A je potvrdený odchod vlaku s presnou hodnotou z hraničnej stanice, avšak príchod vlaku do hraničnej stanici v oblasti B už potvrdený byť nemusí. Ten je dôležitý pre oblasť B, kde tento príchod vieme identifikovať presne. To znamená že skutočné odchody a príchody sa nemusia zhodovať. Preto sme sa rozhodli spájať vlaky v rozdielnych oblastiach pomocou plánovaného odchodu, príchodu, druhu a čísla vlaku.

7.3.2 S počasím

Okrem databázy o prejazdoch vlakov sme získali od SHMU 4 ročnú databázu počasia z mesta Čadca. Tú sme využili a porovnali, či počasie ovplyvňuje meškania vlakov a tiež sme rozšírili vstupnú množinu dát o počasie k zlepšeniu úspešnosti predikcie jazdnej doby.

```
SELECT * FROM [SK-CA] a LEFT JOIN [SK-CA-weather] b on a.DepRealTime > b.dateFrom and a.DepRealTime < b.dateTo
```

Meteorologická stanica v Čadci mala technické problémy a niektoré hodiny neobsahujú informáciu o počasí, preto nám SHMU poslalo tiež záznamy z najbližšej stanice pre možnosť doplnenia chýbajúcich hodnôt. Jedná sa o stanicu Turzovka, ktorá je vzdialená 13 km od Čadce.

7.4 Výpočet pomocných premenných

Pre lepšiu a rýchlejšiu prácu sme si do záznamov dopočítali ďalšie pomocné premenné. Pochádzajú z časových záznamov o plánovaných a skutočných odchodoch a vzdialenosti medzi stanicami, ktoré sme si ručne zistili z internetu pre výpočet priemernej rýchlosti jazdnej súpravy

```
-   [RealDrivingTime] = DATEDIFF(second,DepRealTime, ArrRealTime)
-   [PlanDrivingTime] = DATEDIFF(second,DepPlanTime, ArrPlanTime)
-   [DelayDeparture] = DATEDIFF(second,DepPlanTime, DepRealTime)
-   [DelayArrive] = DATEDIFF(second,ArrPlanTime, ArrRealTime)
-   [Delay] = RealDrivingTime - PlanDrivingTime
```

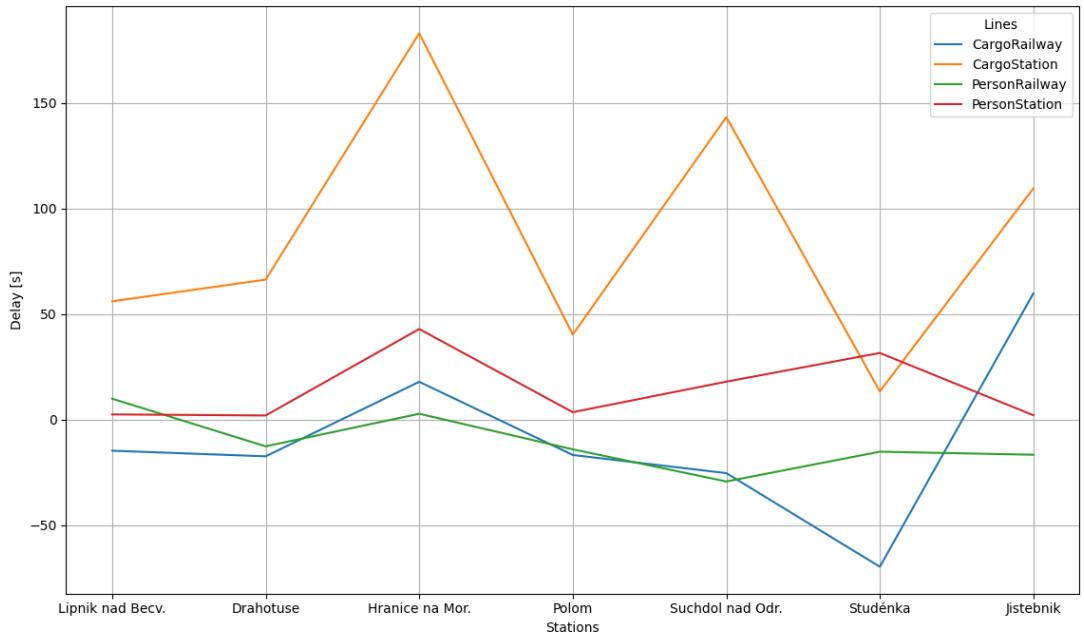
```
-   [planspeed] = 3600*LengthSect/PlanDrivingTime
-   [realspeed] = 3600*LengthSect/RealDrivingTime
```

7.5 Rozbor meškaní

V tejto kapitole sa sústredíme na zdroje meškania a ich závislosti. Niekoľko grafov nám pomôže lepšie porozumieť, kedy tieto meškania vznikajú.

7.5.1 Porovnanie meškaní nákladných a osobných vlakov

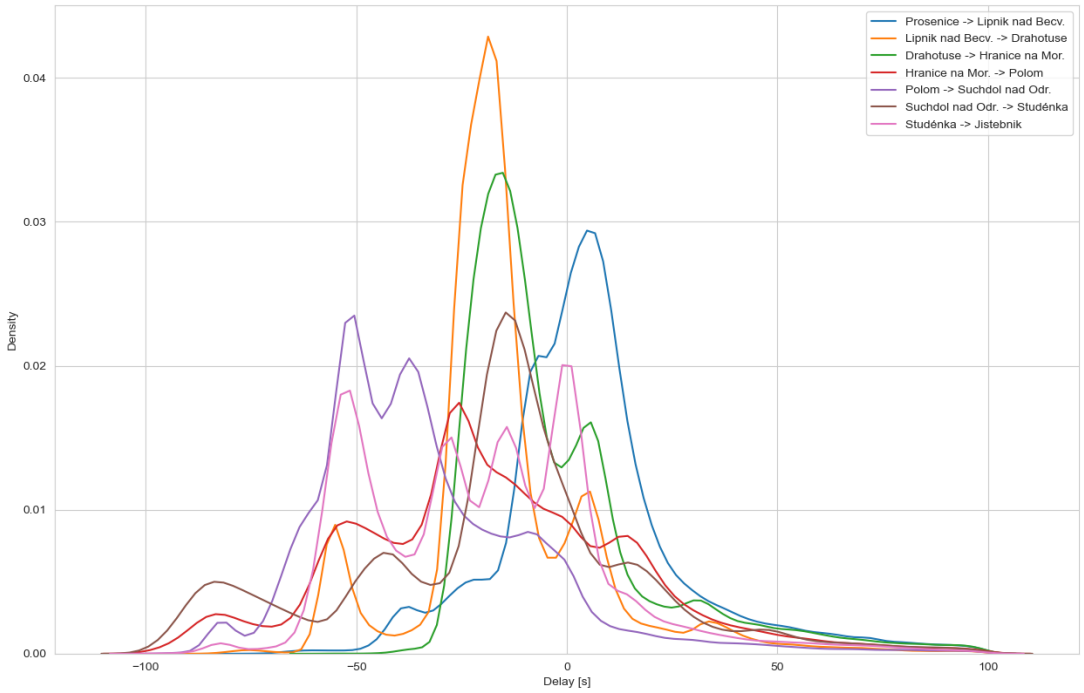
Signifikantný rozdiel v meškaní je medzi nákladnými a osobnými vlakmi a to hlavne v stanici. Tu nákladné vlaky vykazujú výrazne vyššie meškanie a teda predĺžený pobyt. Na obrázku 24 môžeme vidieť porovnanie vzniknutých meškaní na tratiach a staniciach osobitne pre nákladné a osobné vlaky. Ako si môžeme všimnúť, väčšina jazdných dôb nevytvára veľké meškania. V plusových hodnotách sa nachádza meškanie v stanici pre osobné vlaky a ešte vyššie vzniknuté meškanie v staniciach pre nákladné vlaky. Vzniknuté meškania vykazujú závislosti v niektorých staniciach. Napríklad Hranice na Mor. majú zvýšené hodnoty pri všetkých zobrazeniach. Opačne v Studénke v prípade nákladných vlakov je ich meškanie nižšie aj v stanici aj na trati.



Obrázok 24 Meškanie vlakov v medzistaničnom úseku a v staniciach v smere z Lipníka do Jistebníka

7.5.2 Porovnanie meškaní na jednotlivých úsekoch trate

V predchádzajúcom odstavci sme si mohli všimnúť podobné chovanie vlakov v závislosti na stanici. Na obrázku 25 je zobrazené meškanie v staniciach detailnejšie pomocou funkcie hustoty. Môžeme teda vidieť jednotlivé vrcholy a teda najčastejšie sa vyskytujúce meškania na danej trati. Napríklad na trati Studénka Jistebník sú tieto vrcholy až 4 a zároveň je rozptyl výrazne vyšší a ťažší na odhad výsledného meškania.



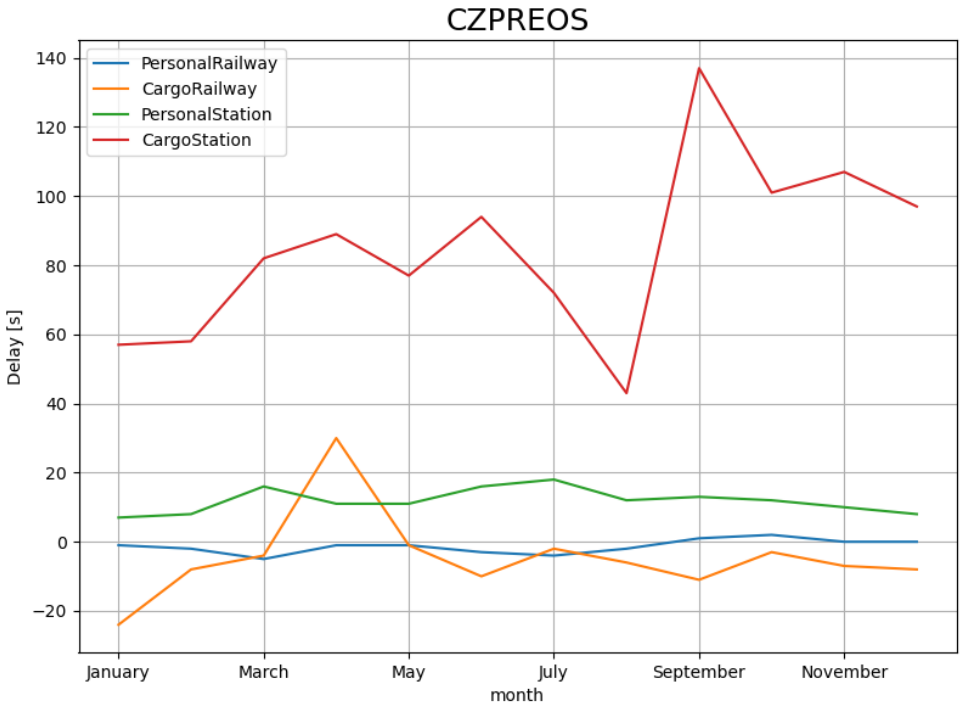
Obrázok 25 Meškanie vlakov na úsekoch

7.5.3 Porovnanie meškaní časove úseky

Analýzu vstupných dát pokračujeme s časom jazdy vlaku. Ten sa pokúsime zhodnotiť z hľadiska časovej granulity po mesiacoch, dňoch a hodinách.

Mesiace

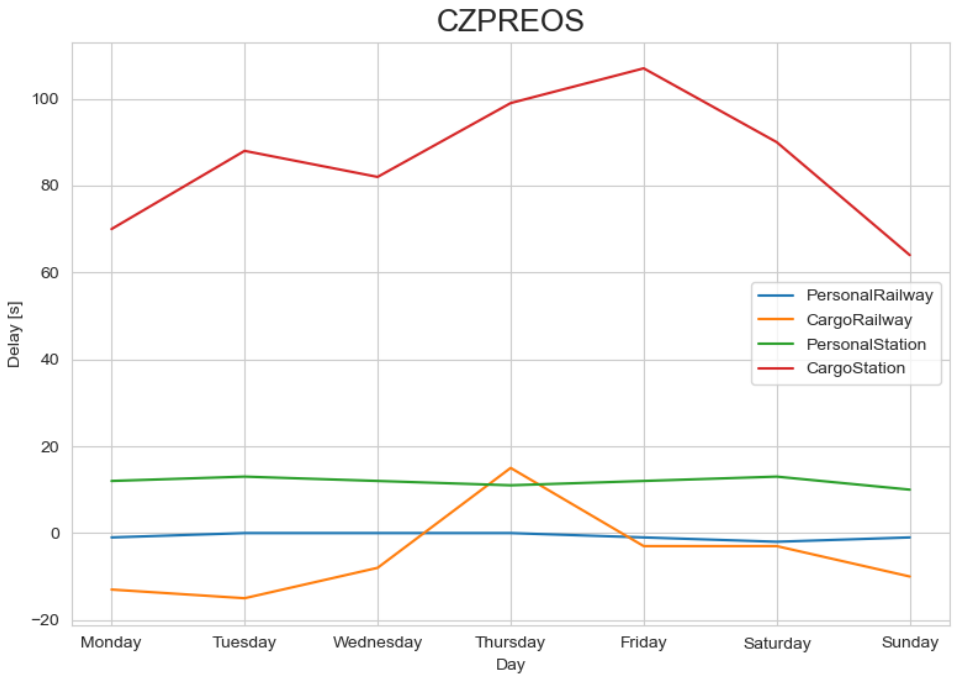
V priebehu roka máme 12 mesiacov, ktoré sú rozdelené do štyroch ročných období. Obzvlášť zimné obdobie môže mať vplyv na jazdnú dobu a na opak, mesiace Júl a August vplyv na osobnú dopravu, kedy žiaci a študenti majú letné prázdniny. Podľa grafu na obrázku 26 vidíme najvýraznejší skok u nákladnej dopravy v mesiaci september. Kedy čas v stanici sa výrazne zvyšuje a súčasne čas jazdy sa mierne znížil. Zvyšné deformácie grafu nie sú tak výrazné a kopírujú trend všetkých mesiacov



Obrázok 26 Závislosť meškania na mesiacoch

Dni

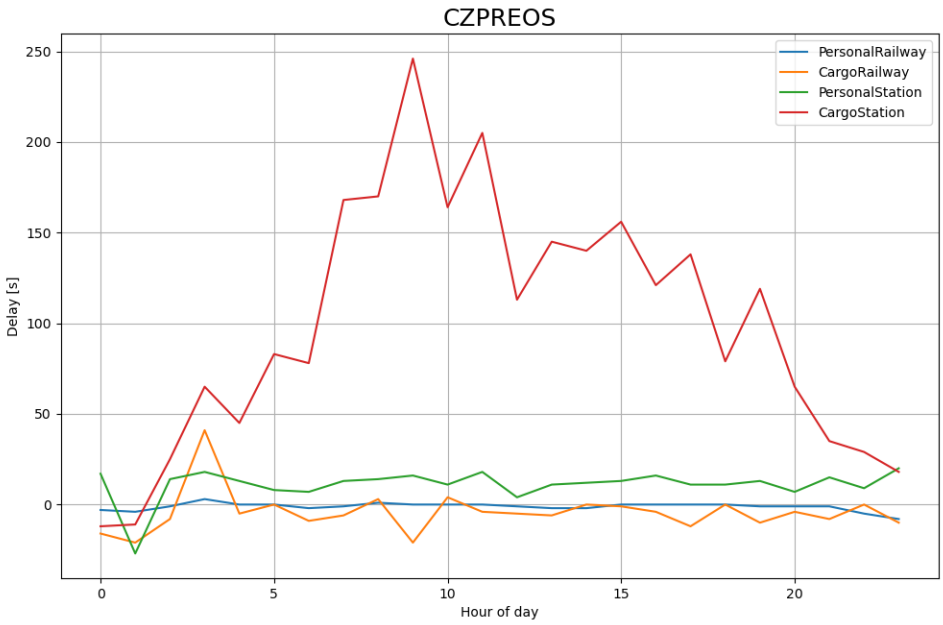
V prípade dní v týždni je opäť najnáchylnejšia nákladná doprava, ktorá od stredy do nedele výrazne spomaľuje jazdu a aj predlžuje pobyt v stanici.



Obrázok 27 Závislosť meškania na dňoch v týždni

Hodiny

Počas dňa vidíme že jazda osobných vlakov veľmi detailne kopíruje hodnotu „0“, teda v priemere spĺňa očakávané hodnoty. V prípade pobytu v stanici týchto vlakov sa čas predlžuje hlavne v ranných hodinách, na obed je pokles a poobede sa opäť vyskytujú meškania, avšak nie tak výrazné ako v ranných hodinách. V prípade nákladných vlakov je pobyt v stanici veľmi často predlžovaný počas dňa, kedy sú koľajnice vyťažené.

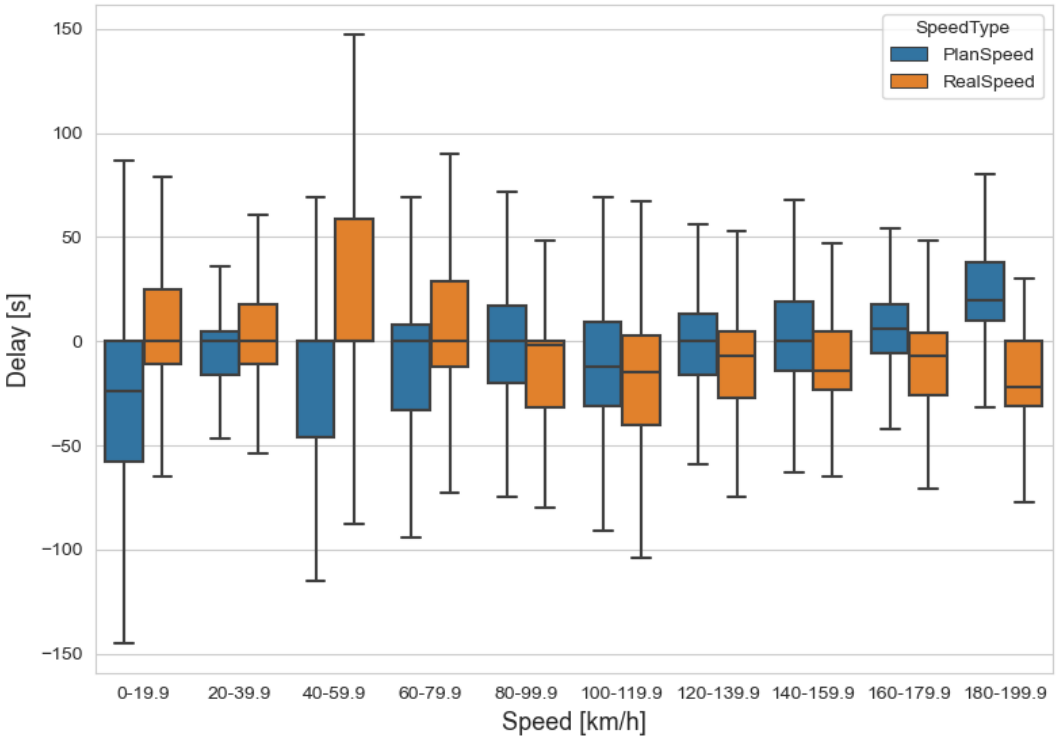


Obrázok 28 Závislosť meškania v priebehu dňa

7.6 Rýchlosť vlaku

Záznamy obsahujú vzdialenosť medzi stanicami a zároveň plánovanú jazdnú dobu. Z toho sa dá vypočítať priemerná rýchlosť vlaku na danom úseku. Samozrejme tento údaj nie je úplne presný, keďže nezapočítavame zrýchľovanie a spomaľovanie vlaku, energetickú efektivitu, stúpanie, klesanie, strmosť zákrut a tiež čiastočné obmedzenia na trati. Je to údaj akoby celá trať bola jednotná vyjadrená priemernou rýchlosťou medzi bodmi. Preto hodnota, ktorá sa bude nachádzať v stĺpci Speed bude o čosi nižšia, ako skutočná maximálna rýchlosť vlaku na trati.

Podľa stránky spravazeleznic.cz, kde je zverejnená železničná trať v Českej republike sa k akruálne nachádzajú úseky len do rýchlosti 160km/h [23]. Po spracovaní a vypočítaní rýchlostí sme označili za chybné záznamy ak rýchlosť presahovala 200km/h. V prípade skutočnej jazdnej doby je to 26736 výskytov a v prípade plánovanej jazdnej doby 23903 výskytov. Väčšina z nich však spadá do oboch skupín a označených je dokopy 28136 záznamov.



Obrázok 29 Závislosť meškania na skutočnej/plánovanej rýchlosti

Graf závislosti je zobrazený na obrázku 29. Skutočná jazdná doba v oranžovom sviečkovom grafe odzrkadľuje výsledok jazdy. Teda ak sa vlak pohybuje v rýchlostiach nad 150km/h, neočakáva sa, že mal problém a teda meškanie by nemalo nastať. Skôr opačne meškanie znižuje. Rovnako to platí v opačnom smere, kde pri priemernej rýchlosti okolo 50km/h sa meškanie výrazne zvyšovalo. Dôležitejšia je pre nás plánovaná jazdná doba zobrazená modrou farbou. Jej trend sa pohybuje nepriamou závislosťou k skutočnej jazdnej dobe. Teda v prípade plánovanej jazdnej doby vo veľkej rýchlosti je veľká pravdepodobnosť že to vlaková súprava nezvládne a tým vznikne meškanie. Potvrdilo sa tiež, že v prípade pomalších úsekov tratí dokážu vlaky svojou energeticky nevýhodnou jazdou, prudkým brzdením či akceleráciou meškanie znížiť. Táto závislosť zaujímavá a pokúsime sa ju použiť aj vo výslednom modeli.

7.7 Rozbor počasia

Najhustejšia a zároveň najväčšia oblasť, ktorú máme je PREOS, teda oblasť medzi Přerovom a Ostravou. Avšak počasie z českej republiky je problémovejšie získať, preto sa v tejto kapitole budeme venovať oblasti Čadca a počasiu, ktoré máme dostupné z tohto mesta. Podobne ako v predchádzajúcej kapitole, sme filtrovanie aplikovali aj na túto

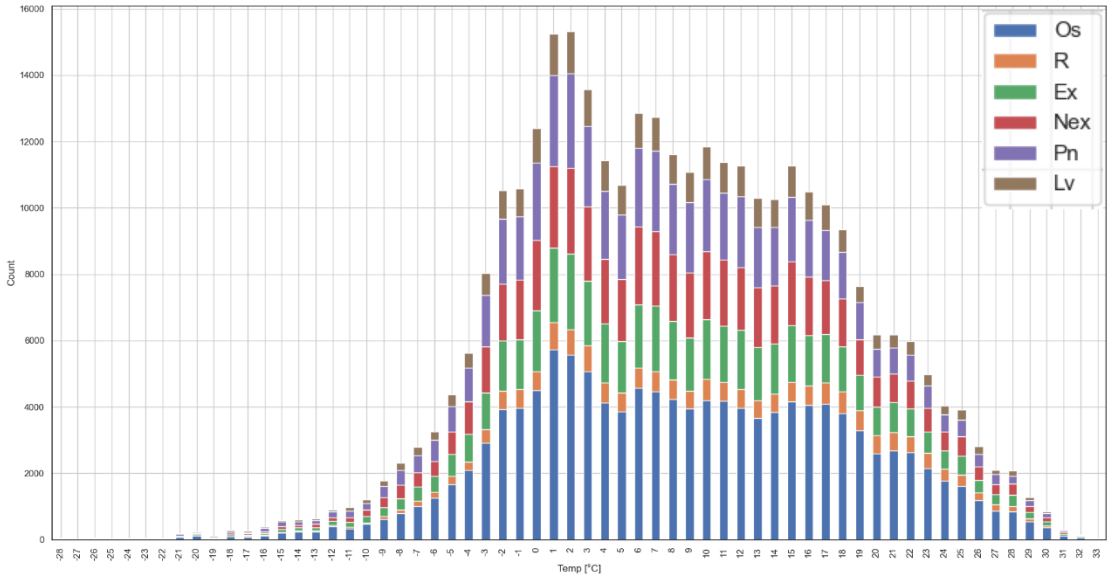
oblasť a ďalej sa jej budeme venovať predovšetkým z pohľadu závislosti počasia na jazdnú dobu. Záznamy obsahujú teplotu, silu a smer vetra, zrážky a zvýšenie snehovej pokrývky.

Slovenský hydrometeorologický ústav nám poslal záznamy z obdobia od 1.4.2016 do 1.4.2020 v časovom intervale jedna hodina. Stanica v Čadci mala niekoľko výpadkov a poskytli nám informácie aj z neďalekej stanice Turzovka na doplnenie celej databázy. Po skombinovaní dát z týchto dvoch staníc nám stále chýbalo niekoľko neznámych hodnôt, kedy obe stanice, nezaznamenávali aktuálne počasie. Ich počet je zobrazený na obrázku 30. Teda napríklad 24.4.2016 je všetkých 24 meraní neplatných.

Den	Pocet
2016-04-23	1
2016-04-24	24
2016-04-25	22
2016-06-15	9
2017-06-29	23
2017-06-30	23
2017-07-01	22
2018-04-22	1
2018-11-13	7
2019-03-21	12

Obrázok 30 Počet chýbajúcich záznamov v jednotlivé dni

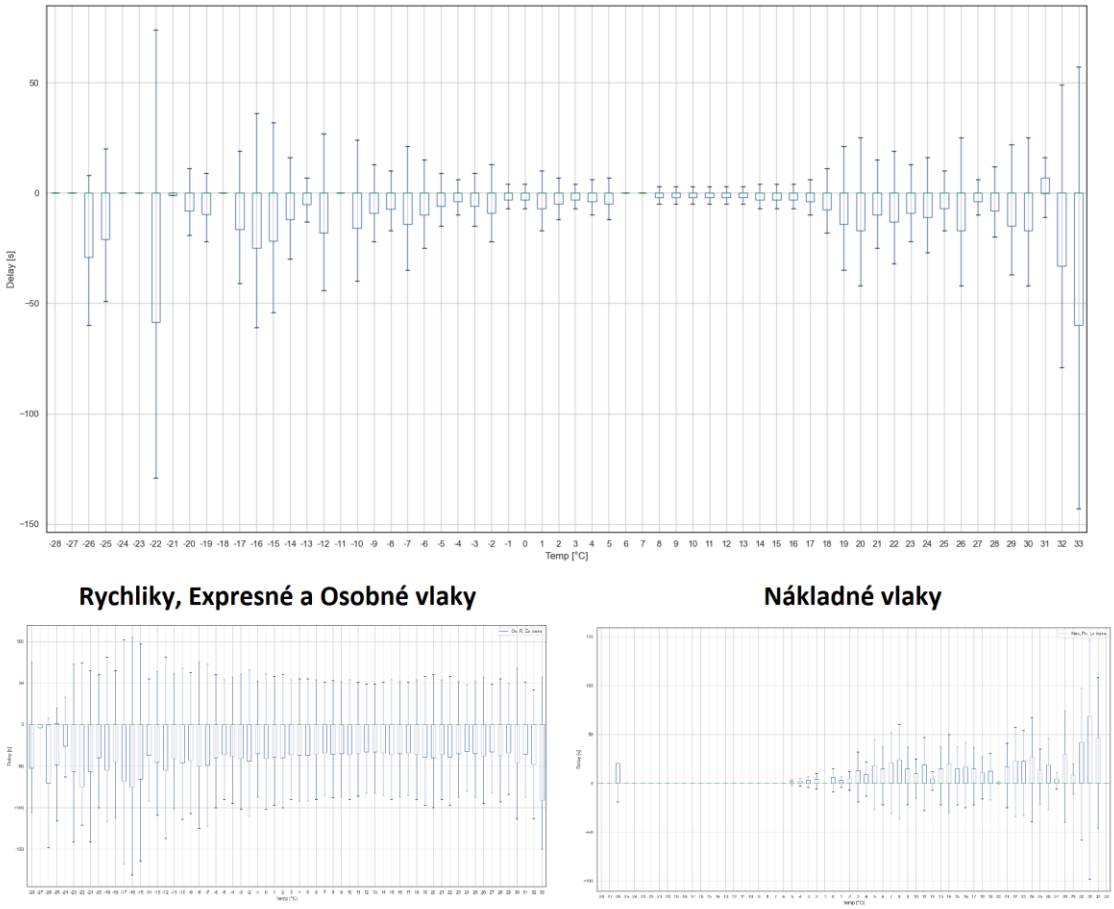
Priemerná teplota v meste Čadca za dané obdobie je 8,37°C avšak vlaky nechodia pravidelne v priebehu dňa. Priemerná teplota jazdy vlaku vychádza na 8,23°C. Môže to byť napríklad frekventovanejšej dopravy v noci. Počty vlakov podľa teploty rozdelené podľa druhu sú zobrazené na obrázku 31.



Obrázok 31 Početnosť vlakov idúcich v uvedenej teplote ovzdušia

7.7.1 Meškanie podľa teploty

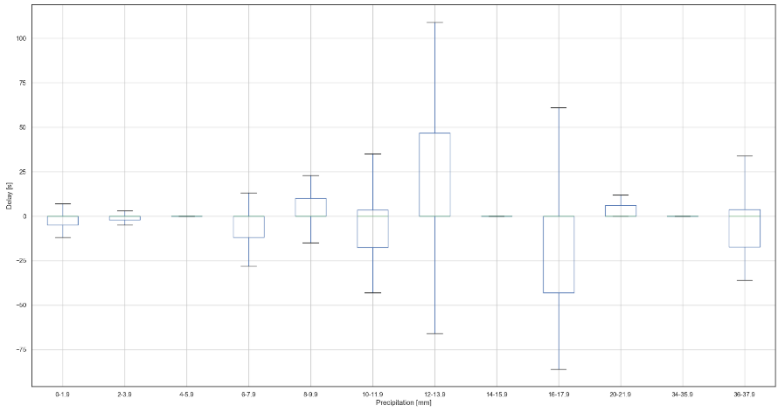
Najočakávanejším vplyvom na jazdnú dobu bolo samotné počasie. Napríklad silné mrazy budú spôsobovať zamrznutie trakčného vedenia a výhybiek. No pri pohľade na grafy na obrázku 32. Vidíme, že ani všeobecný graf ani rozdelenie vlakov na nákladné a osobné neprispelo k závislosti, ktorú sme očakávali. Horizontálna os zobrazuje meškanie, ktoré vzniklo/skrátilo sa na danom úseku. Teda nezahŕňa meškanie, ktoré bolo naakumulované v predchádzajúcich úsekoch. Zaujímavosťou je hlavne rozptyl jazdných dôb pri nízkych a vysokých teplotách. Dá sa teda predpokladať, že hraničné teploty spôsobujú meškania, ale zároveň v prípade meškania vlaku, nie je obmedzený časom a môže jazdu zrýchliť. Čo spôsobí menšiu energetickú efektivitu jazdy, ale podarí sa vlaku skrátiť meškanie.



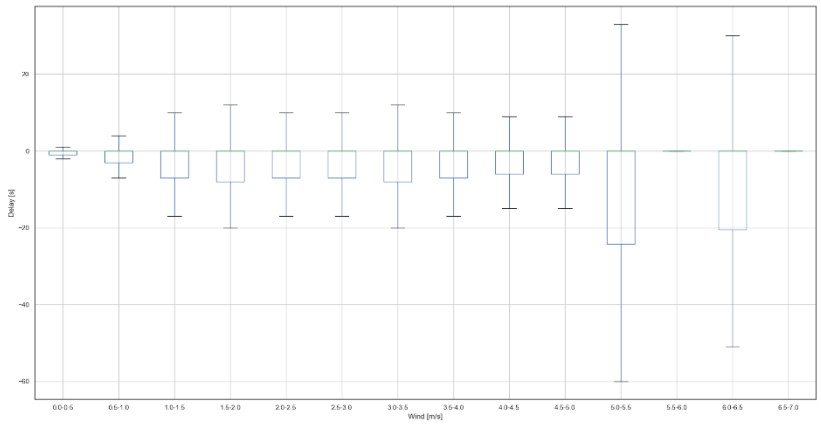
Obrázok 32 Meškanie vlakov podľa teploty

7.7.2 Meškanie podľa ďalších poveternostných podmienok

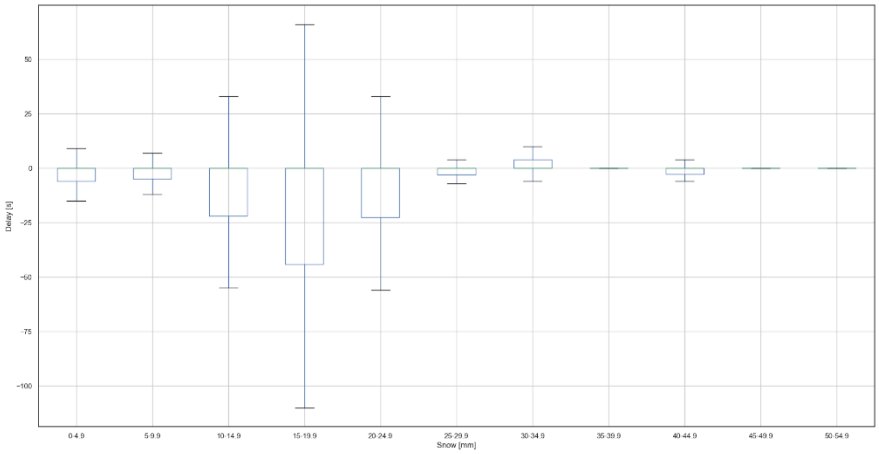
Podobne ako v predchádzajúcej kapitole ohľadom teploty, sme neobjavili ďalšie závislosti v počasí. Na obrázkoch 33 34 a 35 sú zobrazené jednotlivé závislosti, väčšinou hraničné hodnoty spôsobujú meškania, ale zároveň dokážu jazdnú dobu aj skrátiť.



Obrázok 33 Meškanie podľa zrážok



Obrázok 34Meškanie podľa sily vetra



Obrázok 35 Meškanie podľa napadnutého snehu

8 Odhad jazdnej doby

Výstupným parametrom modelu je hodnota jazdnej doby medzi stanicami. Je to udávané ako celé kladné číslo reprezentujúce jazdnú dobu v sekundách a teda reprezentovať ho môžeme ako hodnotu integer. Implementáciu môžeme rozdeliť do niekoľko častí. Začneme **načítaním, spracovaním dát, vybraním metódy učenia, trénovanie, testovanie a hodnotenie dosiahnutých výsledkov**. Tieto kroky budú implementované vo viacerých programovacích jazykoch a algoritmoch umelej inteligencie. Taktiež budú použité viaceré frameworky, ktoré sa pokúsime porovnať. Na záver vytvoríme výslednú aplikáciu, ktorá bude využívať všetky poznatky z analýzy a implementácie k maximalizácii kvality riešenia. Jej funkcionalita bude nasadená ako dotazovacia služba a teda sprístupnená systému GTN a dispečerom, k presnejšiemu odhadu jazdnej doby a príchodu do najbližšej stanice.

8.1 Python s Tensor Flow

K analýze veľkého množstva dát sme využívali jazyk python tak sme s ním pokračovali a využili framework skLearn a Tensor Flow k implementácii prvých modelov umelej inteligencie.

Načítanie dát

K rýchlej práci a adaptácii výpočtov sme sa rozhodli k priamemu načítavaniu dát z databázy bez dočasných súborov. To nám umožní rýchlejšiu selekciu dát a trénovanie na vybranej vzorke dát k vylepšeniu presnosti odhadu. Prácu s databázou MySQL zabezpečuje knižnica *pyodbc*. Umožňuje pomocou definície pripájacieho reťazca vytvoriť spojenie s databázou a následne vykonávať SQL dotazy nad databázou. Keďže SQL podporuje základné operácie s dátami a aby sme predišli problémom s konverzáciou dátových typov v programovacom jazyku, uprednostnili sme vykonať tieto operácie už pri načítavaní aj z dôvodu že implementačný jazyk sa bude meniť a bolo by potrebné všetky dátové transformácie prepisovať. V prípade číselných a kategorických premenných nie sú potrebné žiadne úpravy a programovací jazyk s nimi vie pracovať. Najdôležitejšie úpravy sú v dátumoch, keďže ich reprezentácia sa dá zapísať rôznymi

spôsobmi. Zvolili sme reprezentáciu dátumu ako kombinácie troch čísel. Konkrétne na tri skupiny:

- rok záznamu 2016-2021
- deň v roku s rozsahom 1-366
- sekunda v danom dni s rozsahom 1-86400

Operácie sme vykonali pomocou SQL funkcie DATEPART, ktorá ako argument vyžaduje časť dátumu, ktorý má definovať a premennú, z ktorej má túto hodnotu získať. Príkladom získania sekundy z dňa z premennej „ArrPlanTime“

```
DATEPART(SECOND, [ArrPlanTime]) + 60*DATEPART(MINUTE, [ArrPlanTime]
) + 3600 * DATEPART(HOUR, [ArrPlanTime]) ArrPlanSec
```

Keďže vo vstupných dátach máme štyri časové údaje o plánovanom odchode, príchode a skutočnom odchode a príchode bolo by redundantné získavať rok zo všetkých hodnôt. SQL dotaz SELECT obsahuje informáciu zo všetkých premenných o presnom odchode v sekundách a reprezentácia roku a dňa je získaná len z premennej plánovaného odchodu. Touto operáciou sme síce rozšírili vstupné dáta o dve nové hodnoty, ale získali sme univerzálnu konverziu dátového typu DateTime do číselnej hodnoty.

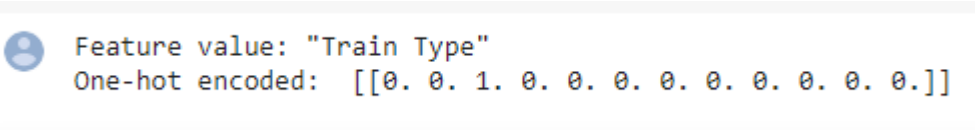
Transformácia kategorických údajov

Umelá inteligencia funguje s číslami, preto je potrebné kategorické premenné ako je typ vlaku, typ motora a identifikácia rušňovodiča prepísať do číselnej podoby. Jazyk python to podporuje pomocou kategorických vlastností a ich kódov. V prípade použitia konvolučných neurónových sieti sme využili príkaz

data["vlastnost"] = data["vlastnost"].astype('category').cat.codes

ktorý nahradíme pôvodné textové reprezentácie kategorických premenných novými číselnými reprezentáciami.

V prípade algoritmu *boosted trees* sa odporúča použitie funkcie *one_hot_cat_column*. Celá implementácia je obsiahnutá vo funkcii frameworku TensorFlow *tf.feature_column.categorical_column_with_vocabulary_list* a jej výstupom je rozdelenie kategorického stĺpca do počtu unikátnych hodnôt. V prípade druhu vlaku, sa tento stĺpec rozšíri na dvanásť stĺpcov (Ex, Lv, Mn, Nex, Os, PMD, Pn, R, Sluz, Sp, Sv, Vlec) a v odpovedajúcom stĺpci sa nachádza hodnota 1, ostatné sú označené hodnotou 0.



Normalizácia premenných

Zlepšenie predikcie dosiahneme normalizáciou premenných, ktorá zarovná rozptyl všetkých hodnôt tak, aby reprezentovali rovnakú dôležitosť. Tensor Flow ponúka implementáciu predspracovania dát normalizačnou vrstvou. Tá využíva výpočet priemeru a rozptylu jednotlivých premenných. Volanie funkcie *Adapt* vypočíta a uloží tieto premenné pre každý vstupný parameter. Následne pred prácou s dátami ako sú *fit, evaluate* a *predict* sa aplikuje normalizácia pomocou vypočítaných hodnôt.

Rozdelenie údajov

V prípade frameworku skLearn vyžaduje rozdelenie vstupných dát, vstupných vlastnosti a výstupnú vlastnosť. A zároveň sme ich rozdelili aj na trénovacie a testovacie v požadovanom pomere. My využívame 15% datasetu na testovanie a zvyšných 85% k trénovaniu.

allCount = len(d)

testCount = int(allCount * testPercent)

x_train = d.head(allCount-testCount).drop(columns='RealDrivingTime')

y_train = d.head(allCount-testCount)['RealDrivingTime']

x_test = d.tail(testCount).drop(columns='RealDrivingTime')

y_test = d.tail(testCount)['RealDrivingTime']

8.1.1 Boosted trees

Implementácia obsahuje definovanie vstupnej funkcie, ktorá je postupne volaná nad menšími časťami dát. Vstupnými parametrami Tensor Flow estimátoru je teda zoznam vstupných stĺpcov, počet opakovaní trénovania nad datasetom a daná vstupná funkcia. Taktiež sa môže obmedziť počet vytvorených rozhodovacích stromov. Prednastavený je limit 100.

tf.estimator.BoostedTreesClassifier(feature_columns,

n_batches_per_layer=n_batches) .train(train_input_fn, max_steps=100)

Po natrénovaní modelu je možné ho uložiť pre neskoršie použitie alebo využiť, kým je načítaný v programe. Funkcia *evaluate* zhodnotí kvalitu modelu, ako moc sa mu podarilo natrénovať na vstupných dátach. Výstup z funkcie *evaluate* je zobrazený na obrázku 36.

```
accuracy                0.850000
accuracy_baseline       0.715000
auc                     0.852667
auc_precision_recall    0.830000
average_loss            0.513541
label/mean              0.320000
loss                    0.410000
precision               0.784386
prediction/mean         0.363636
recall                  0.750000
global_step           100.000000
dtype: float64
```

Obrázok 36 Boosted trees predikcia

Štatistiky tejto predikcie sú len z trénovacieho datasetu. Teda v prípade pretrénovania, dostaneme veľmi presné výsledky, ktoré by však odzrkadľovali realitu. Preto využijeme druhú časť dát určenú na testovanie modelu. Funkciou *predict* odhadneme jazdnú dobu neznámej vlakovej súpravy. Daný výpočet vykonáme na všetkých testovacích dátach a výsledné hodnoty porovnáme so skutočnou jazdnou dobou.

EPOCHS	Absolut Error [s]	SQR Err [s]
2	22,86	35155,93
10	24,67	18701,35
50	26,50	64780,73
100	26,47	88201,62
Planovaná	42,34	1941127,37

Obrázok 37 Boosted trees chybovosť

Na obrázku 37 môžeme vidieť presnosť predikcie nad všetkými dátami s postupným zvyšovaním parametra epochs – počtu opakovaní a porovnaním s plánovanou jazdnou dobou. Absolútna chyba sa výrazne nezmenila, avšak druhá mocnina, ktorá zvýrazňuje zlé odhady druhou mocninou dosiahla najlepší výsledok pri počte opakovaní 10. Následne sa dá usúdiť pretrénovanosť modelu, keďže sa chyba výrazne zvyšuje na neznámych dátach.

8.1.2 Konvolučné neurónové siete

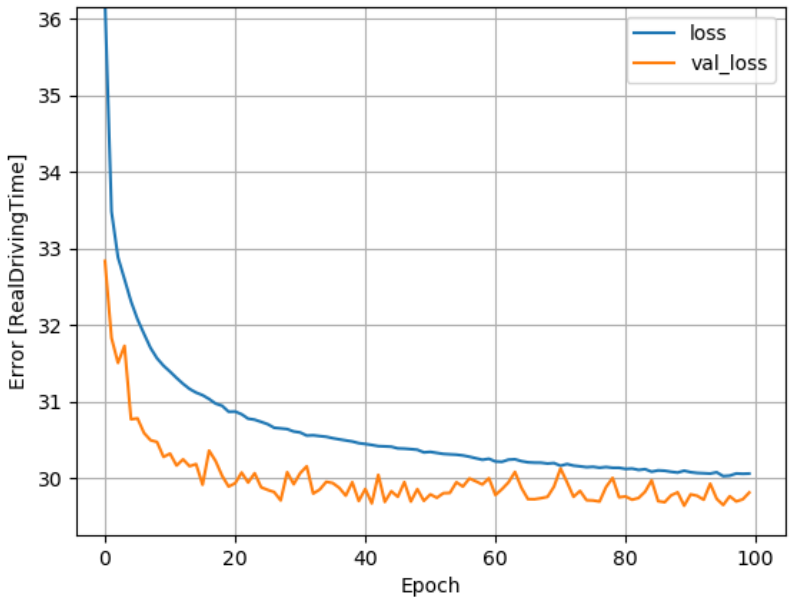
Neurónové siete majú veľkú silu výpočtu a veľké množstvo nastavení pre konkrétny problém. Ich učenie je výrazne náročnejšie a tým zaberá aj viac výpočtového času. Vnútorná štruktúra pozostáva z niekoľkých vrstiev a prepojeniami medzi nimi. Vstupná vrstva obsahuje vstup pre každý vstupný parameter. Následne je trénované vnútorné prepojenie vrstiev a ich vzťahy. Ako východziu konfiguráciu sme zvolili normalizačnú vrstvu a následne tri výpočtové vrstvy. Ako aktivačnú funkciu využíva ReLU (Rectified Linear Unit activation function) ktorej funkcia vykonáva operáciu max(x, 0). Prvá a druhá vrstva obsahuje 64 uzlov, posledná tretia už len jeden výstupný. Podobne, ako pri algoritme *Boosted trees* sme experimentovali s hodnotou opakovaním trénovania nad vstupnými dátami (Epochs). Trénovanie aj testovanie prebehlo nad všetkými dátami a rozdelenie bolo 85/15% na trénovanie a testovanie.

EPOCHS	Absolut Error [s]	SQR Err [s]
2	25,63	31750,59
10	25,88	25319,74
50	27,62	12493,02
100	26,55	15087,16
Planovaná	42,34	1941127,37

Obrázok 38 Konvolučné neurónové siete presnosť odhadu

Ako môžeme vidieť z obrázku 38, pridávaním počtu opakovaní sa absolútna chyba pomaly zvyšovala. Avšak jej rozdiel nie je tak výrazný, ako zníženie druhej mocniny, ktorá dosiahla najlepšie výsledky pri hodnote EPOCHS = 50. Postupný vývoj trénovania hodnôt.

- Loss – reprezentuje presnosť trénovania na známych dátach
- Val_loss – reprezentuje presnosť validácie a tým všeobecnosť modelu na neznámych dátach



Obrázok 39 Konvolučné n. n. - strata informácie pri trénovaní a validácii

Na obrázku 39 je zobrazená závislosť straty informácii pri trénovaní a validácii. V prípade že by sa tieto čiary odďaľovali, naznačovalo by to pretrénovanie modelu. Taktiež ak by oranžová čiara reprezentujúca validáciu obsahovala príliš veľký šum a teda skoky výrazne nad a pod modrú čiaru. Jednalo by sa o nedostatočné množstvo vstupných dát, kde výsledky sú silno viazané na rozdelenie datasetu.

Počet vrstiev

Ďalším parametrom, ktorý sme sa snažili upraviť bol počet použitých vrstiev. Ako nám z predchádzajúceho trénovania vyšlo, najlepšie výsledky boli dosiahnuté pri Epoch okolo 50. Tak sme zvolili hodnotu 60. Z výsledkov na obrázku 40 vidíme, že pridávaním vrstiev sa výsledky nezlepšovali a najlepšou voľbou je ostať na 3 vrstvách.

Layers	Absolut Error	SQR Err	Time [s]
2	24.41	28,716.49	1713.7491
3	26.69	19,608.34	1755.8918
4	27.31	24,133.46	1893.4427
5	27.59	69,939.45	2109.8926
6	28.81	64,803.88	2216.8467
7	28.14	64,719.56	2360.9581
8	28.41	59,813.95	2440.3578

Obrázok 40 Konvolučné n. n. zmena počtu vrstiev

Počet uzlov vo vrstve

Posledným parametrom, ovplyvňujúcim trénovanie a výsledky modelu je počet uzlov v jednotlivých vrstvách. Je možné každú vrstvu nastaviť inak a tým vznikne obrovské množstvo kombinácii. My sme sa pokúšali o uniformné nastavenie všetkých vrstiev rovnako. Na obrázku 41 sú výsledky pri zmene počtu uzlov. Najlepšie výsledky sme dosiahli, pri počte uzlov 64 na vrstvu.

Units	Abs Error	SQR Err	Success	Time [s]
32	26.43	49,405.54	92.50%	1843.5852
48	26.34	40,375.31	92.52%	1832.3817
64	27.31	24,133.46	92.27%	1893.4427
80	28.35	83,214.13	92.00%	1966.413
96	28.94	59,573.44	91.85%	1954.0286
112	28.46	58,507.27	91.97%	2029.7264
128	27.58	61,742.59	92.20%	2223.8114

Obrázok 41 Konvolučné n. n. zmena počtu uzlov vo vrstve

Výsledky predikcii nás doviedli k záveru, že pre náš typ úlohy je najlepšie použitie troch vrstiev s 64 neurónmi na vrstvu a počet trénovaní nad vstupnou množinou s náhodne rozdelenými hodnotami s počtom 60. Tieto nastavenia budú ďalej používané v prípade, že použijeme konvolučné neurónové siete k predikcii skutočnej jazdnej doby. Vstupnými dátami boli všetky vlaky po úpravách a ich priemerná jazdná doba za naše skúmané obdobie dosahuje 5 minút 26 sekúnd. Najlepšie dosiahnutá úspešnosť odhadu jazdnej doby bez použitia selekcie dát je na hodnote 91,62%

8.2 C# s frameworkom ML.NET

Inicializácia

Prvým krokom pri vývoji modelu pomocou frameworku ML.NET je vytvorenie premennej mlContxt reprezentujúca objekt *MLContext* vo funkcii *main*. *MLContxt* tvorí začiatočný príkaz pre všetky ML.NET operácie s umelou inteligenciou. Vytvára sa užívateľom a poskytuje nástroje pre zapisovanie priebehu, monitorovanie výnimiek a nastavenie parametrov náhodného generátora. Je to tiež začiatočný bod pre operácie s

modelom, napr. trénovanie, odhadovanie. Slúži aj ako zoznam dostupných operácií. V našom modeli využívame jeden *MLContext* objekt pre odhad predikcie.

Načítanie dát

Dáta do modelu môžeme načítavať rôznymi metódami. Najjednoduchšie je načítanie dát z textového súboru vo formáte CSV. Vo frameworku ML.NET je zabezpečená operáciou *LoadFromTextFile*. Metóda vracia *IDataView*. *DataView* je základný typ údajovej štruktúry, ktorým dostávame dáta do modelu, jeho základná funkcionalita je porovnateľná s *IEnumerable* pri LINQ.

```
IDataView baseTrainingDataView = mlContext.Data.
    LoadFromTextFile<TrainTrip>(trainDataPath, hasHeader: true, separatorChar: ';');
```

Obrázok 42 Načítanie dát zo súboru

Framework ML.NET načítava dáta podobne ako dotaz SQL. Prvou časťou modelu je objekt, ktorý načíta dáta. V našom prípade načíta súbor dát definovaný textovým názvom s informáciami o jazde vlaku. Objekt sa použije pri vytvorení, natrénovaní a validácii modelu.

Druhým spôsobom načítavania dát o vlakoch do modelu je priame prepojenie s databázou. Často používané načítavanie v frameworku ML.NET je z textového súboru pomocou TextLoader. Jedná sa o lokálne uloženie dát na disku. Avšak z praktického hľadiska bývajú dáta uložené aj inde. V systémoch reálneho času môžu byt uložené v SQL databáze, môžu byť načítané z logovacích súborov alebo dáta vznikajú v čase a postupne sú importované do programu. ML.NET umožňuje zovšeobecnenie schémy a prevziať existujúcu implementáciu C# objektov implementujúcich rozhranie IEnumerable.

```
IDataView dataView = mlContext.Data.LoadFromEnumerable(TrainDatabase.GetOsVlaky());
```

Obrázok 43 Načítanie dát z databázy

Náš program načítava dáta o vlakoch uložené v MS-SQL databáze. O načítavanie a ukladanie dát o vlakoch sa stará rozšírenie Entity Framework (EF) Core. EF Core je aktuálne vyvíjaná, rozšíriteľná, multiplatformová, open-source verzia pôvodnej knižnice Entity Framework Data Access. Technológia spája programové používanie s databázou za pomoci .NET objektov, eliminuje potrebu písania vlastného kódu pre prístup ku dátam uložených v databáze a mapuje relačné databázy do programových objektov. Prístup k dátam sa vykonáva pomocou modelu, pozostáva z kontextového objektu s z tried objektov, ktorý

reprezentuje databázovú reláciu. Vďaka tomuto vzťahu je možné prebrať a ukladať dáta. Objekty sa získavajú z databázy za pomoci jazykovo integrovaných dotazov LINQ – Language Integrated Query. LINQ zastrešuje sadu technológií umožňujúcich prebrať dáta z databázy integrovaných priamo v jazyku C#. LINQ query sa posúva poskytovateľovi databázy preložený do jazyka danej databázy. Dáta o vlakoch sa vytvárajú, mažú a modifikujú v databáze inštanciami objektových tried. Prevádzkovateľ databázy je zodpovedný za vykonanie zmien špecifických operácií databázy, napr. INSERT, UPDATE a DELETE príkazov pri relačných databázach.

```
public static IEnumerable<TrainTrip> GetOsVlaky()
{
    IEnumerable<TrainDB> pomZoznam;
    using (TrainsDbContext db = new TrainsDbContext())
    {
        pomZoznam = db.Trains.Where(t => t.Druh == "Os" && t.GvdOdch!=null && t.GvdPrich!=null).Take(10000).ToList();
    }
    //konvertovanie na train trip
    IEnumerable<TrainTrip> osobneVlaky = converToTrainTripEnumerable(pomZoznam);
    return osobneVlaky;
}
```

Obrázok 44 Import dát z databázy

Transformácia dát

Používanie nespracovaných dát v strojovom učení vedie k nespoľahlivým a zavádzajúcim výsledkom. Analýza a vyčistenie dát je dôležitá súčasť pred tým, ako sa použije súbor dát na trénovanie a testovanie modelu. Keď je dokončený proces trénovania a vyhodnocovania modelu očakáva sa, že hodnoty označenia reprezentujú správny odhad. Keďže je snaha odhadnúť čas cesty jazdnej súpravy, docielime to skopírovaním stĺpca RealDrivingTime do Label stĺpca. Na kopírovanie stĺpcov sme využili transformačnú triedu *CopyColumnsEstimator*. Operácia je na obrázku 45

```
var dataProcessPipeline = mlContext.Transforms
    .CopyColumns(outputColumnName: DefaultColumnNames.Label,
                 inputColumnName: nameof(TrainTrip.JazdnaDoba))
```

Obrázok 45 Kopírovanie označenia v modeli

Spracovanie kategorických premenných

Vstupom do algoritmu strojového učenia, ktorý trénuje model, očakáva ako vstupné vlastnosti len číselné premenné. Preto je potrebné transformovať kategorické dáta (druh vlaku, typ lokomotívy, identifikácia rušňovodiča názov začiatočnej a koncovej stanice) do ich číselnej reprezentácie. Použitím konvertovacej triedy *Microsof.ML.Transorms.OneHotEncodingTransformer* priradíme numerické hodnoty reprezentujúce kategorické premenné.

Normalizácia dát

Algoritmy trénovania očakávajú vstupné vlastnosti v približne rovnakých rozsahoch. V prípade použitia rôznych rozptylov pre jednotlivé vlastnosti môže dôjsť k natrénovaniu suboptimálneho modelu. Taktiež by vstupné hodnoty nemali prekračovať príliš veľké ani príliš malé hodnoty. To vedie k akumulovaniu aritmetickej chyby a tým môže celý proces trénovania zhoršiť. Nami použitá implementácia normalizácie je zobrazená na obrázku 46. Framework ML.NET má viac vstavaných algoritmov normalizácie My sme použili normalizátor *MeanVariance*. Pre každú vstupnú premennú modelu vypočíta smerodajnú odchýlku a strednú hodnotu. Následne lineárne preškáluje hodnoty do intervalu (0,1) tak, aby bola stredná hodnota rovná nule.

```
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.Bod1Cis), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.Bod2Cis), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.Vlak), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.HmotnostVTon), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.Dlzka), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.PocVoznov), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.PocNaprav), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.GvdJazdnaDoba), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.MeskanieOdchod), mode: NormalizerMode.MeanVariance))
.Append(mlContext.Transforms.Normalize(outputColumnName: nameof(TrainTrip.MeskaniePrichod), mode: NormalizerMode.MeanVariance))
```

Obrázok 46 Normalizácia vstupných dát

Zlúčenie vstupných stĺpcov je posledným krokom transformácie dát, ktoré budú použité na trénovanie modelu umelej inteligencie. Vykonáva to transformačná trieda *mlContext.Transforms.Concatenate*. Len stĺpce, ktoré sme v tomto kroku zlúčili, budú použité algoritmom strojového učenia.

```
.Append(mlContext.Transforms.Concatenate(DefaultColumnNames.Features, "Bod1nazEncoded", "Bod2nazEncoded",
"DruhEncoded", "LokomotivaEncoded", "DenPrichoduEncoded", "DenOdchoduEncoded",
nameof(TrainTrip.Bod1Cis), nameof(TrainTrip.Bod2Cis), nameof(TrainTrip.HmotnostVTon), nameof(TrainTrip.Dlzka),
nameof(TrainTrip.PocVoznov), nameof(TrainTrip.PocNaprav), nameof(TrainTrip.GvdJazdnaDoba), nameof(TrainTrip.Vlak),
nameof(TrainTrip.MeskanieOdchod), nameof(TrainTrip.MeskaniePrichod), nameof(TrainTrip.HodinaOdchodu),
nameof(TrainTrip.HodinaPrichodu), nameof(TrainTrip.MesiacPrichodu), nameof(TrainTrip.MesiacOdchodu)));
```

Obrázok 47 Zlúčenie vstupných parametrov

Výber algoritmu učenia

Je mnoho faktorov ovplyvňujúcich výber vhodného algoritmu strojového učenia. Taktiež neexistuje žiadny univerzálny algoritmus, ktorý dosiahne dobré výsledky pri každom probléme. Niektoré problémy si vyžadujú špecifický algoritmus, zatiaľ čo pri iných problémoch môže väčšina algoritmov dosahovať podobne vysokú úspešnosť. Odhad jazdnej doby je numerická premenná, ktorá patrí k regresných problémom. Framework ML.NET disponuje viacerými regresnými algoritmami a my sa pokúsime jednotlivé možnosti porovnať z hľadiska úspešnosti. Ich implementácia je podobná a typ použitého algoritmu sa zadáva ako argument. To nám uľahčuje implementáciu. Dostupné regresné algoritmy frameworku:

- SDCA (Stochastic Dual Coordinate Ascent) Regressor,
- Fast Forest
- Fast Tree.

Trénovanie

Spustenie trénovania zahájime príkazom *Fit*:

```
var trainedModel = trainingPipeline.Fit(trainingDataView);
```

Ktorému ako argument zadáme vstupné dáta k trénovaniu.

Uloženie

Ďalším krokom, kedy už programovo disponujeme natrénovaným modelom. V tejto chvíli sme schopní ho použiť v ľubovoľnej .NET aplikácii. Model využíva kompresiu ZIP formátu pomocou funkcie *SaveModelAsFile* a zadaním cesty, kam sa má uložiť ako argument.

Vyhodnotenie modelu

Spätnou väzbou je úspešnosť predikcie na neznámych dátach, ktoré neboli použité pri trénovaní. K získaniu hodnotenia predikcie sme použili krížovú validáciu, ktorej implementácia je súčasťou framework-u. Pred začatím trénovania modelu sme definovali počet rozdelení datasetu a zahájili sme trénovanie spolu s postupným vyhodnocovaním.

```
var split = mlContext.Regression.TrainTestSplit(data, testFraction: 0.1);
var trainedModel = trainingPipeline.Fit(split.TrainSet);
IDataView predictions = trainedModel.Transform(split.TestSet);
var cvResults = mlContext.Regression.CrossValidate(predictions, trainingPipeline, numFolds: 5);
var r2 = cvResults.Select(r => r.Metrics.RSquared);
var rmse = cvResults.Select(r => r.Metrics.Rms);
```

Použitie modelu

Po úspešnom natrénovaní regresného modelu, môžeme pokračovať v jeho použití a predikcii jazdných dôb aktuálnych vlakových súprav. Prvým krokom k predikcii jazdnej doby, je načítanie modelu z disku.

```
ITransformer trainedModel;
using (var stream = new FileStream(trainedModelPath, FileMode.Open, FileAccess.Read, FileShare.Read))
{
    trainedModel = mlContext.Model.Load(stream);
}
```

Načítaný model má rozhranie *ITransformer*, ktorý ponúka metódu *Transform* k vykonaniu predikcie. V prípade, že máme väčšie množstvo dát, nad ktorými chceme vykonať predikciu. Framework ponúka komponentu *PredictionEngine* ktorá pomocou

dátových tried TrainTrip a TrainTripPrediction vytvorí objekt fungujúcim s textovým súborom, ktorý slúži ako vstup. A jednotlivé riadky reprezentujú vlaky, ktoré chceme predikovať. Trieda TrainTrip reprezentuje parametre vlakovej súpravy a trieda TrainTripPrediction obsahuje predikovanú jazdnú dobu. V tomto prípade nám funkcia vráti pole o dĺžke vstupného súboru a jednotlivé časti poľa reprezentujú odpovedajúcu predikciu zo vstupného súboru.

8.2.1 **Porovnanie algoritmov**

Experimentálne sme spustili rôzne algoritmy umelej inteligencie nad rovnakými dátami a v tabuľke 8 sú zobrazené dosiahnuté výsledky. Najlepšie hodnoty dosiahol algoritmus FastTree a zároveň s najkratším časom trénovania sa ukázal ako najlepšia voľba frameworku ML.NET.

Algoritmus	Absolut Error	SQR Err	Success	Time [s]
FastTree	24.42	27,643.48	93.03%	40.08
SDCA	38.17	72,468.12	89.52%	42.93
FastForest	25.82	23,468.23	92.66%	42.9

Tabuľka 8 Porovnanie algoritmov

8.2.2 **Porovnanie frameworkov**

Opísané boli frameworky ML.NET v jazyku C# a TensorFlow jazyka python. Ich ponuka algoritmov umelej inteligencie sa líši a tým sme získali aj rozdielne úspešnosti predikcie. V prípade programovacieho jazyka Python je implementácia o niečo prehľadnejšia a jednoduchšia. Python patrí medzi skriptovacie jazyky, čím ponúka aj výhodu pri práci s „big data“ teda rozsiahlymi datasetmi, čo je v našom prípade výhodou. Vytvorený program je prekladaný za behu, čo znamená že chybný kód je odhalený počas jeho vykonávania a tým čiastočne komplikuje vývoj. V prípade C# je pred spustením skompilovaný celý program do strojového jazyka a až po tom spustená jeho reprezentácia v strojovom kóde. Je taktiež silno typový a všetky konverzie typov sú skontrolované počas kompilácie. Najvyššiu úspešnosť predikcie v jazyku C# sme dosiahli 93,09% v porovnaní s konvolučnými sieťami v jazyku Python 93,03%

Zložitosť implementácie

Jazyk python je stručnejší a tým aj kód je prehľadnejší. Zároveň ponúka dostatok možností parametrov trénovania aj testovania. Vďaka voľnejšej typovosti, je jednoduchšie pretypovať kategorické premenné do ich číselnej reprezentácie. Definícia vstupnej trénovacej funkcie jazyka python je zobrazená na obrázku 48. Obrázok tiež zobrazuje možnosti modifikácie vstupnej premennej, nastavenie počtu epochs pomocou Tensor Flow estimatora.

```
def get_input_fn(data_set, num_epochs=None, n_batch=128, shuffle=True):
    return tf.estimator.inputs.pandas_input_fn(
        x=pd.DataFrame({k: data_set[k].values for k in FEATURES}),
        y=pd.Series(data_set[LABEL].values),
        batch_size=n_batch,
        num_epochs=num_epochs,
        shuffle=shuffle)
```

Obrázok 48 Vstupná funkcia (python)

HW/SW náročnosť – cena

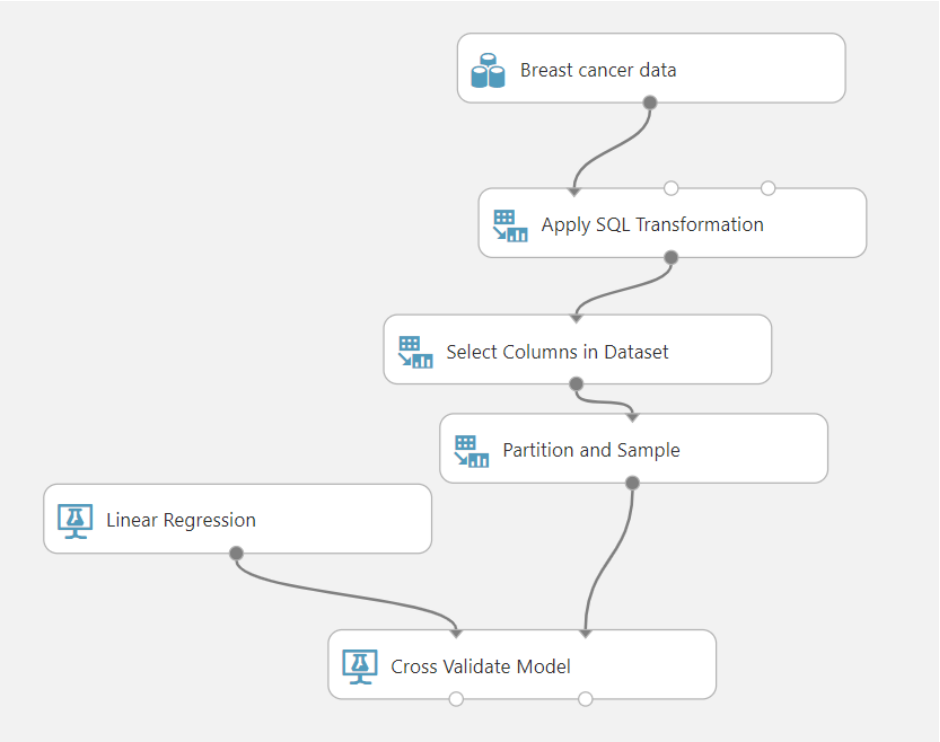
Spomenuté riešenia sú vytvorené na pracovnom počítači so systémom Windows. V prípade programu v ML.NET je táto technológia podporovaná .NET Frameworkom, ktorý je viazaný na operačný systém windows, ale aj .NET Core, ktorý je multiplatformový. Takže výsledný program je možné použiť aj na počítačoch či serveroch, založených na linuxovom jadre. V prípade python je situácia jednoduchšia, keďže jazyk je od základov podporovaný multiplatformovo, problém s migráciou a spustením na rozdielnych platformách by nemal byť problém. Hardvérová náročnosť je v tomto prípade porovnateľná. V prípade licencii je ML.NET publikovaný pod MIT licenciou. V prípade TensorFlow sa jedná o Apache License 2.0.

8.3 Odhad jazdnej doby cloudovým riešením

Zaujímavým riešením je využitie cloudového priestoru. Model umelej inteligencie sme implementovali v prostredí od Microsft (Azure machine Learning Studio) a od Google (Google Datalab).

Microsoft

Spoločnosť Microsoftu využíva k implementácii Azure machine Learning Studio. Poskytuje komplexný webový balík strojového učenia a dátovej analýzy. Od Google datalabu sa značne odlišuje, kde bolo nutné ručne písať príkazy. Vo vývojovom prostredí od Microsoftu nemusíme písať kód a môžeme využiť jeho interaktívne rozhranie založené na moduloch, kde každý modul zastrešuje určitú funkciu. Aj neskúsený užívateľ zvláda vytvoriť pokročilé modely strojového učenia vďaka priateľskému a prehľadnému užívateľskému prostrediu. Program je možné rozšíriť pomocou modulov, ktoré vedia spracovať R alebo Python skript. Implementáciu Lineárnej regresie vo webovom prostredí od firmy Microsoft je zobrazená na obrázku 49.



Obrázok 49 Lineárna regresia v Azure Machine Learning studio

Google

Cloudové riešenie od spoločnosti Google sa volá Cloud Datalab. Je interaktívnym a výkonným nástrojom, vytvoreným na analýzu, transformáciu a vizualizáciu údajov. Ponúka tiež vytváranie modelov strojového učenia. Pri registrácii v prostredí Datalabu si musí užívateľ vytvoriť virtuálny stroj o definovanom výpočtovom výkone na ktorom bude vykonávať svoju prácu. Podľa výkonu záleží aj suma, ktoré sa bude účtovať za využívanie služby. Ďalej prostredie Datalab ponúka možnosť využiť Google BigQuery,

je založený na SQL databáze a uľahčuje prácu s dátami. Skripty sú písané v jazyku Python a tým ponúka aj všetky knižnice dostupné v tomto jazyku.

Porovnanie

V oboch Cloudových riešeniach sme implementovali algoritmus lineárnej regresie nad rovnakými dátami. Oba cloudové systémy túto úlohu zvládli s rovnakou úspešnosťou a rozdiely vo výsledkoch boli menej ako 1%. Riešenie od Microsoftu je z hľadiska implementácie výrazne jednoduchšie, prehľadnejšie a dá sa zvládnuť aj bez hlbšej znalosti jazyka. Google je kombináciou programovacieho a grafického rozhrania čím je prehľadné vďaka grafickej časti a zároveň mu pridáva širšie možnosti implementácie v programovacej časti s možnosťou použitia knižníc. Jedinou nevýhodou týchto riešení je ich rýchli vývoj a časté zmeny. Tým že je to webový vývoj, nie je možné pracovať so starou verziou, ale núti nás to používať zmenený dizajn. Taktiež vývoj umelej inteligencie prináša časté zmeny a tým spojená menšia popularita a absencia dobrej dokumentácie a návodov nadšencov a vývojárov využívajúci túto platformu. Celkovou výhodou cloudových riešení je ponuka ceny len za to, čo skutočne program použije. Hlavnou výhodou je v prípade nárazovo veľkých nárokov na výpočtový výkon, ktorý v prípade fázy učenia by sa dal využiť. Následné načítanie modelu a predikcia už nepotrebuje toľko výpočtového výkonu.

8.4 Zlepšenie predikcie selekciou dát

Vstupné dáta, na ktorých je predikcia závislá sme spracovali. Vyskúšali sme aj viaceré učiace sa algoritmy. Avšak ako zobrazujú viaceré grafy v analýze, dáta sú rôznorodé a správanie vlakov sa dosť líši aj podľa ich typu. Preto sa pokúsime dáta rozdeliť a pre každú skupinu vytvoriť vlastný model umelej inteligencie. To znamená, že výsledný program by obsahoval niekoľko modelov a na základe vlaku, pre ktorý sa má vypočítať jazdná doba by program identifikoval, do ktorej skupiny patrí a podľa toho použil príslušný model. Výpočty sme robili na spracovaných vstupných dátach. Použili sme konvolučné siete s 4 vrstvami a 64 uzlami na vrstvu, keďže dosiahli najlepšie hodnotenie predikcie.

Typ vlaku

Prvým rozdeľovacím kritériom sme zvolili typ vlaku. V databáze sa nachádza 12 unikátnych typov ako môžeme vidieť aj v tabuľke 2, čo by nám však rozdelilo vstupné dáta na moc malé časti a tým by trénovanie nemuselo dosiahnuť kvalitné výsledky. Spôsobom jazdy je výrazný hlavne medzi nákladnými, osobnými a servisnými vlakmi. Rozhodli sme sa pre rozdelenie do troch skupín.

- Nákladné (Nex, Pn, Mn, Lv)
- Osobné (R, Os, Ex)
- Ostatné (PMD, Vlec, Sp, Sv, Sluz)

TrainType	Count	Plan Abs Err	Plan Sqr Err	Plan Succes	Pred Abs Error	Pred Sqr Err	Time [s]	Pred Succ
Os Ex R	2936651	28.87	9,059.87	90.92%	17.9	784.43	6020	94.17%
Nex Lv Mn Pn	2123540	69.22	1,081,267.88	84.45%	41.35	134,165.43	4267	90.09%
PMD Vlec Sp Sv Sluz	161312	54.86	18,835.60	85.86%	36.09	5,089.92	314	90.22%

Tabuľka 9 Predikcia podľa typu vlaku

V tabuľke 9 môžeme vidieť dosiahnuté výsledky predikcie pri rozdelení dát podľa typu vlaku. Osobné vlaky sa nám podarilo predikovať výrazne lepšie. Vzhľadom že osobné vlaky zastrešujú väčšiu časť dát, vážený priemer chyby celkového modelu by dosiahol lepšiu hodnotu po rozdelení datasetu, ako keď sme používali celú vzorku dát a vytvorili jeden model. Pri pohľade na úspešnosť sa každý model zlepšil o 5% k plánovanej jazdnej dobe.

Časové obdobie

Dátum skrýva v sebe niekoľko možností delenia. Rozhodli sme sa vyskúšať rozdeliť záznamy podľa hodiny odjazdu, dňa v týždni a ročného obdobia. Využívame pri tom SQL funkciu DATEPART, ktorá. ako argument používa typ rozdelenia. Konkrétne hour, weekday a month reprezentujúce hodinu, deň v týždni a mesiac.

Time	Count	Plan Abs Err	PlanSQR Err	Plan Succ	Pred Abs Err	Pred Sqr Err	Pred Succ	Time [s]
0-6	1,043,121	41.59	26,417.89	89.13%	26.76	2,428.19	92.72%	2263
6-12	1,325,095	41.56	65,696.25	88.57%	23.62	2,501.84	93.17%	2771
12-18	1,573,082	38.82	28,101.24	89.06%	23.41	1,924.88	93.10%	3377
18-24	1,141,336	61.94	1,673,456.52	84.12%	28.87	13,272.11	91.91%	2430

Season	Count	Plan Abs Err	PlanSQR Err	Plan Succ	Pred Abs Err	Pred Sqr Err	Pred Succ	Time [s]
Winter	1,285,561	39.64	19,337.57	89.04%	27.33	10,831.29	92.18%	2767
Spring	1,258,790	38.62	22,237.56	89.52%	24.43	1,719.35	93.11%	2729
Summer	1,211,688	42.83	53,457.59	88.26%	25.09	1,960.87	92.77%	2606
Autumn	1,465,464	60.01	1,511,103.40	84.61%	49.36	1,485,155.53	86.99%	3200

WeekDay	Count	Plan Abs Err	PlanSQR Err	Plan Succ	Pred Abs Err	Pred Sqr Err	Pred Succ	Time [s]
Mo, Fri	1,512,198.00	47.19	529,700.31	87.32%	25.95	45,164.33	92.61%	2926
Tu, We, Th	2,341,386.00	48.46	615,631.39	87.20%	24.27	2,795.70	93.15%	5005
Sa, Su	1,367,919.00	40.88	60,849.69	88.67%	26.43	45,066.48	92.37%	2618

Tabuľka 10 Predikcia podľa časového obdobia

V prvej časti tabuľky 10 sú časovo rozdelené dni podľa hodín do štyroch skupín. Podľa počtu záznamov vidíme, že najmenej vlakov premáva po polnoci do skorého rána a najviac v popoludňajších hodinách. Z pohľadu meškania sa výrazne odlišujú len večerné vlaky, u ktorých výrazne stúpla hlavne štvorcová chyba. To naznačuje vznik menej častých ale väčších meškaní. V prípade ročných období sa opäť ukazuje jeden z dát ako problémovejší. Tým je jeseň, kedy výrazne stúpla absolútna aj štvorcová chyba. Tu si s ňou však model umelej inteligencie nevedel moc poradiť a nezlepšil výsledky tak výrazne ako v prípadne rozdelenia podľa časov. Poslednou časťou je rozdelenie týždňa na pracovný začiatok a koniec, stred týždňa a víkend. Rozdelenie hustoty dát je rovnomerné, meškania vznikajú približne rovnako počas celého týždňa. Rozdiel je v dĺžke meškania. Počas týždňa zvyknú meškania dosahovať väčšiu hodnotu na rozdiel od víkendov, kedy sú meškania síce častejšie ale nie tak výrazné. Vidíme to na porovnaní plánovanej absolútnej chyby s plánovanou štvorcovou chybou.

Z pohľadu časového rozdelenia sa nám podarilo najviac zlepšiť predikciu vo večerných hodinách, u ktorých sa znížila absolútna aj štvorcová chyba. Na druhú stranu odhad jazdnej doby v jeseni nepriniesol výrazne zlepšenie ani v jednom z hodnotiacich parametrov.

Úsek trate

Železničná riadená oblasť obsahuje 23 železničných tratí. Ako reprezentatívnu vzorku sme vybrali 8 náhodných tratí tak, aby sme vedeli porovnať kvalitu predikcie.

Po zoradení hustoty premávky na jednotlivých úsekoch sme vybrali veľmi frekventované ale aj menej využívané úseky. Keďže sme použili selekciu dát, tak sa zmení aj chyba plánovanej jazdnej doby. V tabuľke 11 je náhodný výber úsekov a smeru s počtom záznamov v danom smere a následne absolútna a štvorcová chyba plánovanej jazdnej doby a predikovanej spolu s časom trénovania. Ako môžeme vidieť, úspešnosť sa výrazne zlepšila. Obzvlášť štvorcová chyba je výrazne nižšia, čo znamená, že modelu sa darí predikovať vzniknuté väčšie meškania. Najmenšia chyba je na trati Prosenice -> Přerov, kde absolútna chyba plánovanej jazdy je len 0,5sekundy. Reprezentuje priemernú chybu. V takomto prípade je veľmi ťažko konkurovať tak presnému odhadu. Predikčný model zvýšil absolútnu chybu, ale podarilo sa mu výrazne znížiť štvorcovú chybu, ktorá je jeho optimalizačným parametrom pri trénovaní.

Railway	Count	Plan Abs Err	PlanSQR Err	Plan Succ	Pred Abs Err	Pred SQR Err	Time [s]	PredSucc
Hranice na Mor. -> Drahotuse	240589	39.76	8,481.25	85.74%	25.78	1,647.93	468	90.26%
Jistebnik -> Studenka	231449	39.59	28,608.11	88.87%	20.32	1,432.54	475	93.96%
Prosenice -> Lipnik nad Becv.	218091	45.52	38,388.27	87.58%	25.97	2,279.83	445	92.52%
Studenka -> Jistebnik	212733	41.2	14,696.70	87.85%	27.8	1,947.98	410	91.47%
Polom -> Hranice na Mor.	205860	59.83	15,247.58	87.79%	34.03	4,181.67	435	92.67%
Prosenice -> Prerov os.n.	134887	0.5	113.8	99.88%	0.92	1.71	258	99.77%
Prerov os.n. -> Prosenice	123662	86.64	78,748.26	83.86%	63.13	5,805.70	266	87.70%
Bilovec -> Studenka	24211	47.63	6,316.06	93.83%	44.91	4,465.35	49.5379	94.16%

Tabuľka 11 Odhad jazdnej doby na jednotlivých úsekoch trate

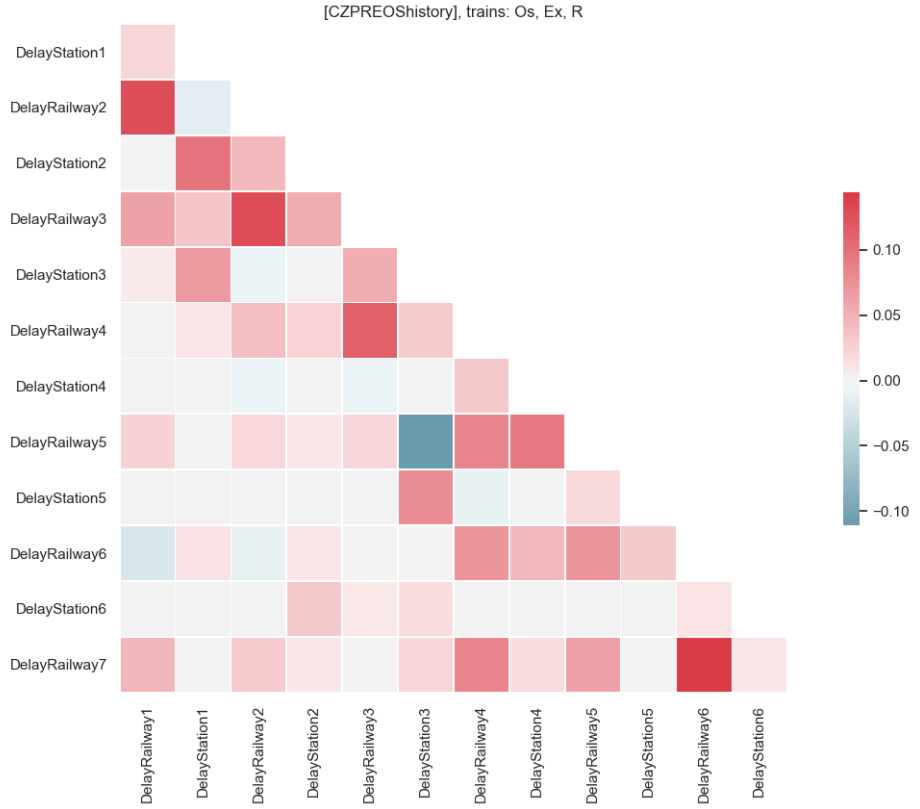
Experimentálne sme sa pokúsili zamerať na jeden úsek trate s tým, že modelu pridáme informácie aj o iných úsekoch. Očakávali sme zlepšenie predikcie, keďže model má možnosť naučiť sa správanie vlaku aj na iných úsekoch a tým zlepšiť predikciu na našom skúmanom. Informácie z iných úsekov sú len ako druhotná informácie a preto pre zdôraznenie dôležitosti dát zo skúmaného úseku, sme tieto dáta duplikovali. Konkrétne 9 násobne. Tým sme vytvorili pomer 9:1 dát zo skúmaného úseku k dátam z iných úsekoch. Tento experiment nepriniesol pozitívne výsledky a veľmi sa podobal modelu, ktorý pracoval nad všetkými dátami.

8.5 Zlepšenie odhadu teoretickej jazdnej doby využitím predchádzajúcich úsekov

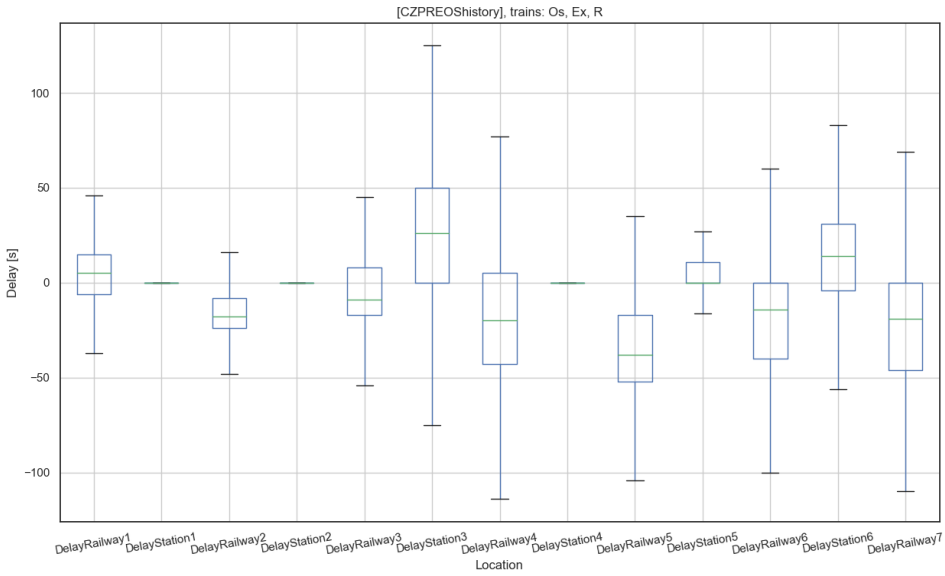
V predchádzajúcich kapitolách sme sa snažili predpovedať skutočnú jazdnú dobu na základe historických dát a s využitím umelej inteligencie. Výpočet teda nie je závislí na aktuálnej situácii. V tejto kapitole sa pokúsime korigovať teoretickú (predpokladanú)

jazdnú dobu na základe aktuálnych vlastnostiach jazdnej súpravy. Využijeme historické dáta k stanoveniu parametrov výpočtu jazdnej doby na aktuálnom medzistaničnom úseku pomocou znalosti jazdných dôb daného vlaku v predchádzajúcich medzistaničných úsekoch.

Využijeme riadenú oblasť PREOS a jej najdlhšiu hlavnú vetvu medzi Přerovom a Ostravou. Konkrétne sa zameriame na odhad medzistaničného úseku Jistebník Polanka s využitím informácii z 6 úsekov pred ním. Mapa danej oblasti je zobrazená na obrázku 13. Pôvodnú databázu prejazdov sme transformovali do záznamov, ktoré obsahujú skutočné a plánované časy odjazdov a príjazdov s indexom 1-7, o ktorý úsek sa jedná. Ďalej obsahujú výpočty o meškaniach na trati (railway), stanici (station) a oneskorení príjazdu a odjazdu. V prípade nákladných vlakov, ktorých je väčšina sú závislosti nejasné. Spôsobené nižším dôrazom na ich meškanie ako u osobných, kde už niekoľko minútové meškanie je významné. Zobrazenie závislosti meškaní počas jazdy je v korelačnej matici na obrázku 50. Korelačná matica zobrazuje líniu vyššej závislosti na diagonále posunutej o jedno nižšie ako je hlavná. Tá zobrazuje závislosť meškania osobitne v medzistaničnom úseku a v stanici. Zaujímavá je záporná závislosť medzi meškaním v stanici 3 a medzistaničnom úseku 5, ktorá pri doplnení informácii z obrázka 51 vysvetľuje časté meškanie v stanici 3 Hranice na Morave. Toto je kompenzované na úseku 5 (Polom-Suchdol), kde vlaky zvládajú skrátiť jazdnú dobu a tým znížiť aj meškanie.



Obrázok 50 Korelačná matica meškaní počas jazdy



Obrázok 51 Graf meškaní vlaku počas jazdy

Pre nás najdôležitejší parameter je DelayRailway7, ktorý znázorňuje meškanie pri príjazde do našej sledovanej stanice Jistebník. Reprezentuje pre nás výstupnú premennú a korelácia s ostatnými vstupnými parametrami je žiadúca. Meškanie s výraznou

pozitívnou koreláciou je s úsekom 6. Táto vlastnosť veľmi pomôže, keďže daný úsek je blízko sledovaného a aj hodnoty meškania na obrázku 51 sú podobné.

Konkrétne hodnoty korelácie meškania na medzistaničnom úseku 7 Studénka Jistebník (DelayRailway7) sú v tabuľke 12.

Názov	Lokácia	Korelácia	Lin. Regresia
DelayRailway1	Prosenice -> Lipnik nad Becv.	0.046696	0.051042
DelayStation1	Lipnik nad Becv.	0.005454	0.003837
DelayRailway2	Lipnik nad Becv. -> Drahotuse	0.030352	0.035443
DelayStation2	Drahotuse	0.009914	0.00144
DelayRailway3	Drahotuse -> Hranice na Mor.	0.003861	-0.01674
DelayStation3	Hranice na Mor.	0.022233	0.004262
DelayRailway4	Hranice na Mor. -> Polom	0.085933	0.057053
DelayStation4	Polom	0.016095	0.003276
DelayRailway5	Polom -> Suchdol nad Odr.	0.064208	0.049011
DelayStation5	Suchdol nad Odr.	0.005543	-0.00059
DelayRailway6	Suchdol nad Odr. -> Studénka	0.144372	0.076061
DelayStation6	Studénka	0.010273	0.000371

Tabuľka 12 Korelačne a regresné hodnoty jazdy vlaku

K výpočtu očakávaného meškania na medzistaničnom úseku číslo 7 sme použili Lineárnu regresiu. Vstupom boli meškania daného vlaku v predchádzajúcich úsekoch a staniciach, teda model nemal k dispozícii informácie o vlakovej súprave, typu vlaku, váhy a iných dostupných dát. Výpočet bol teda len z informácii o jeho jazdnej dobe. Konkrétne hodnoty výpočtu cez lineárnu regresiu sú v poslednom stĺpci v tabuľke 12. Väčšinou kopírujú korelačnú maticu. Avšak v korelačnej matici boli použité všetky dáta a v lineárnej regresii sme použili rozdelenie 75% k 25%. Teda zdroj hodnôt je 75% dát a zvyšných 25% sme využili na testovanie a hodnotenie kvality výpočtu. Zaujímavosťou je, že ani korelácia ani regresia nepreukázali závislosť jazdnej doby na dobu strávenú v stanici. Takže sa dá predpokladať, že väčšinou ide o celkovú poruchu a zníženie rýchlosti jazdy v celej riadenej oblasti a neovplyvňuje dobu v stanici.

Druhý model predikcie sme vytvorili použitím algoritmu neurónových sieti. Vstupné dáta sú rovnaké a reprezentujú meškania na doterajšej trati na jednotlivých úsekoch a staniciach. Nastavenia neurónovej siete a počet vrstiev sme použili rovnaké, ako v predchádzajúcej kapitole. Výsledky sú zobrazené v tabuľke 13. Prekvapivo plánovaný čas má najmenšiu štvorcovú chybu. Avšak jeho priemerná absolútna chyba dosahuje vysokú hodnotu. Čo znamená, že priemerne sa skutočná jazdná doba líši o 40 sekúnd.

algorithm	Absolut err	SQR Err
plan time	40.05	13,961.66
Linear r.	30.15	15,864.84
Neural network	20.87	24,423.53

Tabuľka 13 Predikcia podľa aktuálnej jazdenej doby

8.6 Kombinácia odhadov

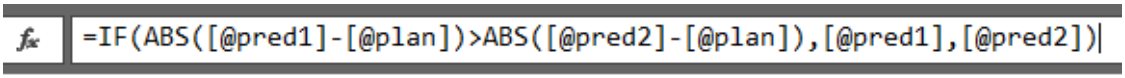
V predchádzajúcich kapitolách sme vytvorili model umelej inteligencie, ktorý sa naučil odhadovať jazdnú dobu pomocou historických dát a vedomostí z nich získaných. Jedná sa teda o off-line predikciu, ktorá nepotrebuje prísun nových dát a odhaduje na základe zadaných parametrov vlaku. Parametre odhadu sú označené ako „TrainSet spec“. Tento model by sme chceli použiť v kombinácii s on-line odhadom z kapitole 8.5, kde ako vstupné informácie potrebujeme aktuálne dosiahnuté jazdné doby vlakovej súpravy. Reprezentuje ho riadok „Track delays“ v tabuľke 14. Prvý riadok reprezentuje plánovanú jazdnú dobu. Ďalšie dva reprezentujú výsledky predchádzajúcich modelov tentokrát na menšej vzorke. Konkrétnej trati s 163780 záznamami.

Druhá polovica tabuľky reprezentuje spojenie dvoch odhadov, kde sme vyskúšali tri matematické funkcie. Podľa výsledkov je predikcia na základe vlastností vlaku výrazne lepšia v oboch parametrov avšak aj online predpoveď z aktuálne dosiahnutých meškaní reprezentuje určité zlepšenie. Preto sa pokúsime získať výhody z oboch a tým vytvoriť ešte lepšie odhady. Tou najjednoduchšou funkciou je priemer dvoch časových odhadov v tabuľke reprezentovaný ako „Mean“. Hodnotu absolútnej chyby si udržalo na veľmi nízkej úrovni a štvorcová chyba sa tiež moc nezhoršila. Avšak oba parametre sú o čosi horšie ako v prípade „TrainSet spec“. Druhou funkciou sme vypočítali vzdialenosť spomenutých dvoch predikcií od plánovanej jazdnej doby a ako výstupnú hodnotu sme použili tú vzdialenejšiu v prípade algoritmu „HighestDiff“ a bližšiu v algoritme „LowestDiff.

Algorithm	Abs Error	SQR Error	Success
Plan time	34.70	21,811.70	85.61%
Track delays	29.40	21,504.62	90.37%
TrainSet spec	18.46	10,492.40	93.73%
Mean	24.66	14,581.46	91.80%
HighestDiff	16.43	9,707.20	94.38%
LowestDiff	30.04	21,289.83	90.19%
Records count: 163780			

Tabuľka 14 Kombinácia predikcie

V prípade použitia funkcie, ktorá použila odhad viac podobajúci sa plánovanej jazdnej doby sa absolútna chyba ešte zhoršila a štvorcová chyba ostala na hodnote horšej predikcie. Tým je tento spôsob nepoužiteľný a len zhoršuje odhad. No v prípade opačného postupu, kde vyberáme za výstupnú hodnotu tú vzdialenejšiu hodnotu sa nám podarilo dokonca znížiť absolútnu chybu. Spájanie dvoch predikcii sme implementovali v programe Microsoft Excel, kde sme si vložili do tabuľky všetky záznamy, ich plánovanú, skutočnú jazdnú dobu a obe predikcie. Následne sme vypočítali „HighestDiff“ pomocou vzorca na obrázku 52.



Obrázok 52 Vzorec kombinujúci predikcie

8.7 Spätný odhad parametrov jazdnej súpravy

Počas analýzy a spracovávaní výsledkov sme pôvodnú vstupnú databázu s 5 264 496 záznamov redukovali na 5 221 507 čím sme síce prišli o 42 989 (8,17%) dát. Ale ich kvalita vzrástla výrazne. Vypovedá o tom aj zlepšenie modelu predpovede. Preto sa v tejto kapitole zameriame na spätnú kontrolu parametrov vlaku. Pokúsime sa na základe dosiahnutých jazdných dôb odhadnúť parametre vlaku a porovnať so zadanými hodnotami. V prípade dosiahnutia dobrých výsledkov by sme vedeli rozšíriť program GTN o kontrolu parametrov, ktoré sú získavané z ISOŘ a následne upozorniť na ich dôveryhodnosť. Možností implementácie a kombinácii vstupných dát je veľké množstvo. Skúsime začať s najmenšou vstupnou informáciou a tou bude skutočná jazdná doba.

Ďalej ju skúsime rozšíriť o plánovanú jazdnú dobu prípadne zvolíme niektoré z parametrov za dôveryhodné k lepšiemu overeniu iných.

8.7.1 Odhad parametrov

Začneme s odhadom typu vlaku. Je to kategorická premenná a v skúmanej riadenej oblasti nadobúda 10 unikátnych hodnôt. V prípade použitia kategorickej premennej je možné ju zakódovať do číselnej reprezentácie alebo použiť „one_hot_encoding“. Teda reprezentovať každú unikátnu hodnotu ako stĺpec s binárnou hodnotou.

TrainId	RealDrivingTime	TrainType
1821076	141	Ex
1821084	321	Pn
1821093	250	Pn
1821108	137	Ex
1821121	188	R
1821124	217	Pn
1821127	138	Ex
1821130	154	Ex
1821131	270	Os
1821136	122	Ex

Ex	Os	Pn	R
1	0	0	0
0	0	1	0
0	0	1	0
1	0	0	0
0	0	0	1
0	0	1	0
1	0	0	0
1	0	0	0
0	1	0	0
1	0	0	0

Obrázok 53 Kódovanie kategorickej premennej TrainType

V prvom príklade je typ vlaku výstupnou premennou. Vstupom pre trénovanie modelu je len hodnota skutočnej jazdnej doby ktorú vlak dosiahol. Identifikačné číslo vlaku je zobrazené len pre kontrolu správnosti výpočtov a nefiguruje v procese trénovania a odhadovania modelu. Nezvyčajný je počet výstupných parametrov, ktorý je v skrátenej ukážke o hodnote 4 avšak skutočný model ich obsahuje až 10.

Algoritmus lineárnej regresie nespadá do klasifikačných metód, preto použijeme konvolučné siete, podobné ako sme používali aj v predchádzajúcich modeloch. Rovnako tak použijeme už existujúce funkcie k načítavaniu dát priamo z databázy pomocou SQL dotazov. Východziu konfiguráciu sme zvolili na úrovni 20 opakovaní a použitie dvoch vrstiev. V tabuľke 15 môžeme vidieť úspešnosť tohto modelu pri zmene počtu neurónov (uzlov) vo vrstvách. V tretom stĺpci je zobrazená hodnota „Accuracy“ reprezentujúca presnosť modelu. Teda koľko záznamov sa modelu podarilo správne odhadnúť len pomocou vedomosti o skutočnej jazdnej dobe. Najvyššie sú zobrazené najúspešnejšie modeli. Ako môžeme vidieť, modelu neprospieva, keď je rovnaký počet neurónov v oboch vrstvách.

20 Epochs, 10 categories		
1. layer	2. layer	Accuracy
16	64	54.65%
64	32	53.36%
8	32	53.10%
32	16	52.49%
64	64	52.40%
32	64	52.38%
8	16	51.84%
16	32	51.77%
32	8	51.73%
16	8	50.29%
8	8	48.86%
64	16	48.49%
16	16	48.37%
32	32	48.22%
64	8	29.17%

Tabuľka 15 Vplyv počtu neurónov na spätný odhad typu vlaku

Ďalším nastavovaním parametrov trénovania sme zvolili počet vrstiev. Na základe znalostí z predchádzajúcich výsledkov, sme sa snažili pri zvyšovaní vrstiev nastaviť adekvátne aj počet uzlov v nich. Pri väčšom počte vrstiev sme experimentálne vyskúšali aj viaceré možnosti. Ako môžeme vidieť v tabuľke 16, najhoršie výsledky dosiahol model s jednou vrstvou. Dve vrstvy priniesli mierne zlepšenie a pri použití troch a viac vrstiev boli výsledky veľmi porovnateľné. Trénovanie je robené postupným náhodným rozdeľovaním dát a trénovaním v niekoľkých krokoch. Preto presnosť odhadu sa môže líšiť pri viacnásobnou spustení algoritmu a nemôžeme hodnotiť model za lepší, ak dosiahol vyššiu presnosť o jedno či menej percent.

20 Epochs, 10 categories						
1. layer	2. layer	3. layer	4. layer	5. layer	Accuracy	Time [s]
16	-------	-------	-------	-------	48.14%	245.0
16	64	-------	-------	-------	54.35%	348.6
16	32	64	-------	-------	54.75%	334.4
8	16	32	64	-------	54.36%	383.9
8	32	16	64	-------	54.11%	432.9
8	16	32	16	64	54.39%	462.3
8	32	16	32	64	54.31%	473.2
16	32	16	32	64	54.47%	480.9
32	16	32	16	64	54.40%	469.9
64	16	32	16	64	55.08%	435.5
64	32	16	32	64	54.03%	486.9
64	16	64	16	64	54.56%	462.7
64	32	64	32	64	53.86%	480.5

Tabuľka 16 Vplyv počtu vrstiev na spätný odhad typu vlaku

Po nastavení vrstiev, ich počtu a definície sme vyskúšali vylepšiť predikciu ešte pomocou počtu opakovaní. Zvolením príliš veľkého čísla sa dostaneme do stavu pretrénovania, pri použití malého počtu opakovaní sa náš model nedostatočne natrénuje. Do teraz sme používali nastavenie EPOCHS = 20. V tabuľke 17 je zobrazená presnosť odhadu na základe počtu opakovaní vstupných dát v trénovanej fáze. Trénovaná bola neurónová sieť s troma vrstvami a počtom neurónov 16,32,64 v jednotlivých vrstvách. Spolu s počtom opakovaní sa adekvátne zvyšuje aj dĺžka trénovania modelu. Presnosť odhadu za zvyšuje po hodnotu 20 opakovaní a následne drží približne rovnakú presnosť. Z toho usudzujeme, že zvýšenie opakovaní nad 60 v trénovanej fáze nám zvyšuje čas trénovania bez výrazného zlepšenia predikcie.

layers unit 16,32,64		
Epochs	Accuracy	Time [s]
1	41.80%	51.7
5	51.47%	97.8
10	52.97%	181.1
20	54.35%	334.4
40	54.48%	719.7
60	54.56%	1043.3
100	53.72%	1503.9
200	53.27%	2430.1
300	54.61%	2932.9
500	54.07%	4491.2
1000	55.01%	9191.6

Tabuľka 17 Vplyv hodnoty Epochs na spätný odhad typu vlaku

Do teraz sme sa venovali a modifikovali predikčný model k dosiahnutiu čo najpresnejších výsledkov len pomocou jednej vstupnej premennej skutočnej jazdnej doby. V ďalšom kroku sme rozšírili vstupnú premennú aj o plánovanú jazdnú dobu. Tabuľka 18 reprezentuje úspešnosť predikcie modelu pre rôzne nastavenia počtu vrstiev a neurónov v nich. Presnosť odhadu sa úmerne zvýšila pridaním plánovanej jazdnej doby do vstupnej množiny o približne 10%. Čas trénovania ostal približne na rovnakej úrovni. Jedná sa o čas zodpovedajúci len trénovaniu modelu a celkový čas behu programu je výrazne dlhší, keďže v prvej fáze sa načítavajú dáta z externej databázy. V tomto prípade načítanie dát trvalo okolo 20 sekúnd, keďže trénovanie sme spúšťali na serveri, ktorý obsahuje aj databázu.

60 Epochs, input: Real and Plan driving time						
1. layer	2. layer	3. layer	4. layer	5. layer	Accuracy	Time [s]
16	-------	-------	-------	-------	53.62%	591.75
16	64	-------	-------	-------	60.29%	678.6
16	32	64	-------	-------	62.28%	734.58
8	16	32	64	-------	61.94%	821.58
8	32	16	64	-------	62.32%	899.49
8	16	32	16	64	62.52%	927.99
8	32	16	32	64	63.02%	906.81
16	32	16	32	64	62.86%	895.17
32	16	32	16	64	62.57%	866.73
64	16	32	16	64	62.53%	856.95

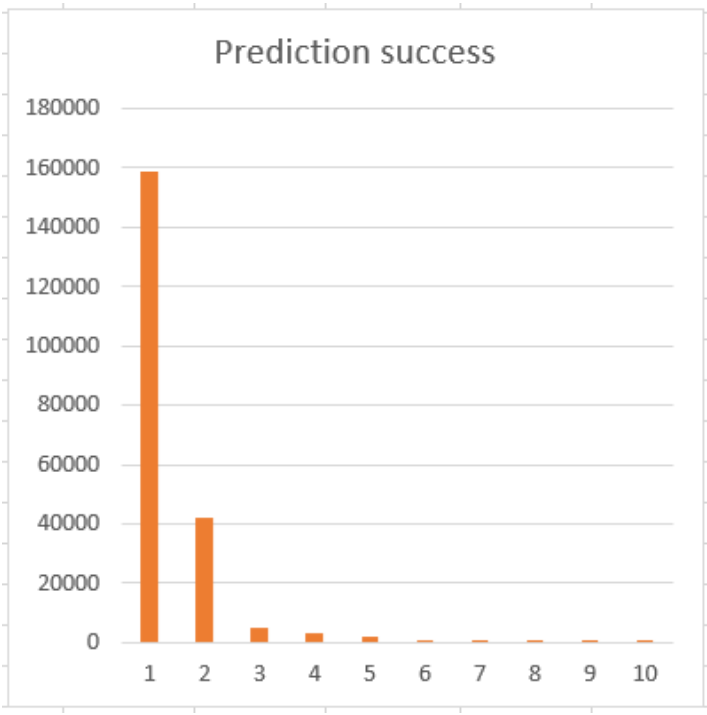
Tabuľka 18 Presnosť predikcie druhu vlku s dvomi vstupnými parametrami

Rozširovať vstupnú množinu môžeme ďalej fyzikálnymi vlastnosťami vlaku, teda jeho váhou, dĺžkou, počtu vagónov a náprav. Keďže sú tieto vlastnosti navzájom závislé a pochádzajú z rovnakého zdroja, použili sme ako reprezentatívnu vlastnosť jeho váhu, ktorá ma najväčší rozptyl hodnôt. Samozrejme očakávame ďalšie zlepšenie odhadu, avšak spoliehame sa na správnosť zadaných údajov. V prípade skutočnej jazdnej doby sú zdrojom zabezpečovacie systémy na tratiach, plánovanú jazdnú dobu získavame z informačného systému Kango a druh vlaku sme kontrolovali zo systému ISOŘ. Tým sme prepojili tri systémy a vzájomne kontrolovali správnosť údajov. Pridaním váhy vlaku, ktorá je získavaná z ISOŘ pre pravidelné vlaky spolu s jeho typom už zvyšuje závislosť spoľahlivosti na jeden z troch zdrojov. V tabuľke 19 sme získali ďalších 20% úspešnosti odhadu kategorickej premennej Typ vlaku vďaka rozšíreniu vstupných premenných o váhu vlakovej súpravy. Rozšírením vstupného vektoru sa zvýšil aj čas trénovania a dokázali sme odhadovať správne až 80% vlakov. A to stále pri výbere 1 kategórie z celkových 10 možností.

60 Epochs, input: Weight, Real and Plan driving time						
1. layer	2. layer	3. layer	4. layer	5. layer	Accuracy	Time [s]
16	-------	-------	-------	-------	78.09%	899.25
16	64	-------	-------	-------	78.35%	1078.27
16	32	64	-------	-------	81.71%	1239.87
8	16	32	64	-------	82.95%	1298.34
8	32	16	64	-------	82.35%	1320.36
8	16	32	16	64	82.86%	1342.97
8	32	16	32	64	83.63%	1328.57
16	32	16	32	64	82.46%	1335.41
32	16	32	16	64	82.41%	1304.34
64	16	32	16	64	82.78%	1242.75

Tabuľka 19 Presnosť predikcie druhu vlku s tromi vstupnými parametrami

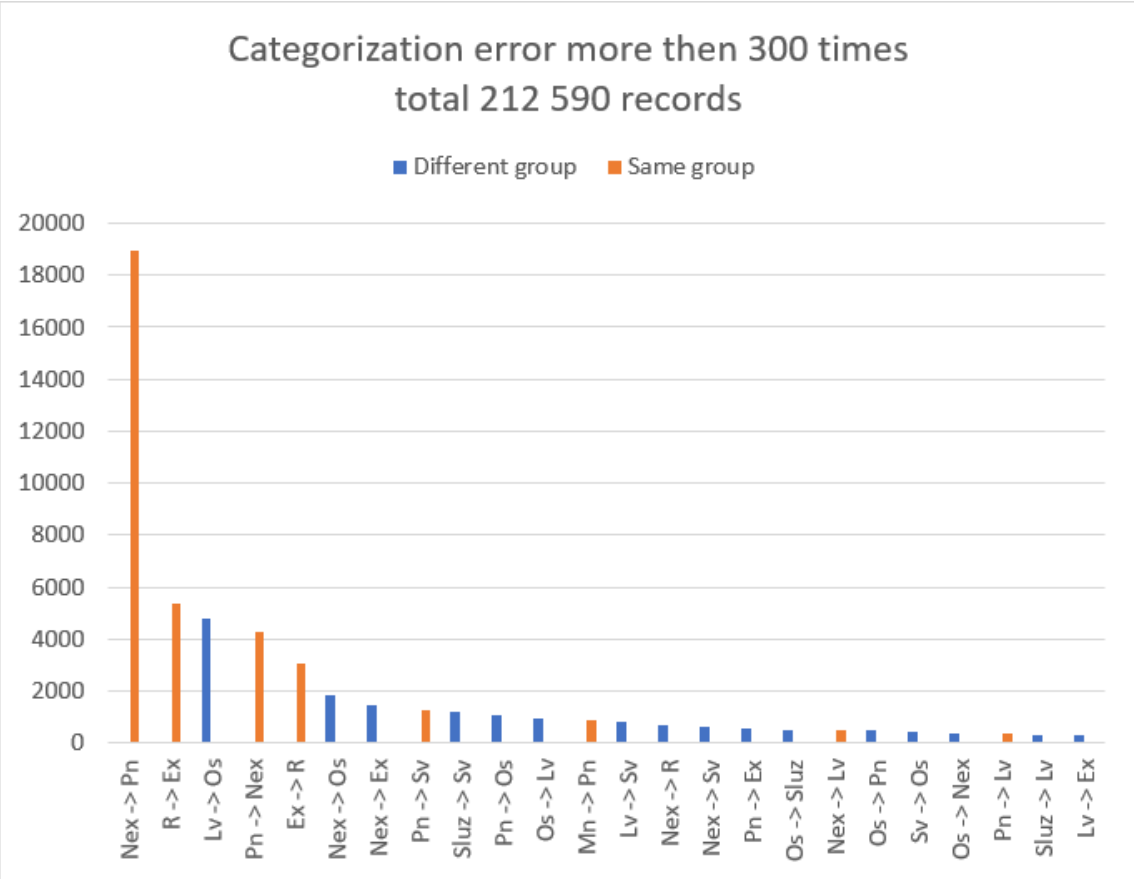
Výstupný parameter predikcie je dvojrozmerné pole. Kde jednotlivé riadky reprezentujú odhady a stĺpce všetky druhy vlakov, ktoré premenná môže nadobúdať. Pole obsahuje desatinné čísla od 0-1 reprezentujúce pravdepodobnosť daného typu zo zadaných parametrov. To znamená že súčet čísel každého riadku je 1 a pravdepodobnosti sú rozdelené medzi jednotlivé typy. Vďaka tomuto vieme predikovať konkrétny typ vlaku, prípade ak chceme zvýšiť presnosť odhadu, môžeme znížiť počet druhov zjednotením. Keďže si ukladáme všetky predikcie a pravdepodobnosti jednotlivých typov. Vieme vypočítať úspešnosť najpravdepodobnejšieho odhadu ale aj napríklad dvoch či troch najpravdepodobnejších. Na obrázku 54 je zobrazená táto úspešnosť. V prípade že očakávame od modelu presný druh vlaku, jeho úspešnosť bude 74,75%. V ďalších 19,68% správne odhadol typ vlaku na druhý krát a tak ďalej. Ak by program akceptoval informáciu, že vlaková sústava bude jeden z dvoch typov vlakov, vieme dosiahnuť úspešnosť až 94,43%. Tieto dva druhy vlaku nie sú na seba viazané, je to len inteligencia modelu, ktorá jeho správanie zhodnotila že sa podobá na dva druhy, ktoré sa nemusia zdať podobné.



Order	Count	Percent
1	158914	74,75%
2	41846	19,68%
3	5260	2,47%
4	3332	1,57%
5	2024	0,95%
6	799	0,38%
7	299	0,14%
8	81	0,04%
9	26	0,01%
10	9	0,00%
sum	**212590**	

Obrázok 54 Úspešnosť pokusov odhadu typu vlakovej súpravy

Predchádzajúca možnosť odhadovania dvoch typov vlaku, nám nezaručuje že tieto typy budú rovnakého druhu. Na obrázku 55 sú zobrazené najčastejšie chyby v odhadoch. Ako môžeme vidieť, najčastejšie označil model vlak „Nex“ (nákladný expres) ako vlak typu „Pn“ (Priebežný nákladný). Ich počet je zobrazený oranžovou farbou, reprezentujúcu chybu v rovnakej kategórii. Ako bolo opísané rozdelenie na nákladné, osobné a ostatné vlaky v kapitole 8.4. Prvou modrou hodnotou je označenie vlakovej súpravy Lv ako Os, čo reprezentuje rozdielnu skupinu vlakov. Konkrétne sa to vyskytlo v 4775 prípadoch, čo reprezentuje chybu v 2,25%. V súčte pri použití rozdelenia podľa typu vlaku do troch skupín sme dosiahli pravdepodobnosť úspechu 91,27 %.



Obrázok 55 Početnosť nesprávne predikovaných typov vlakovej súpravy

8.7.2 Odhad numerických parametrov vlaku

Medzi numerické parametre vlaku, ktoré je možné odhadnúť patrí skupina závislých premenných váha, dĺžka počet náprav a vagónov. Podobne ako pri kategorických premenných sa pokúsime odhadnúť numerické premenné len pomocou dosiahnutej skutočnej jazdnej doby, s pridaním plánovanej jazdnej doby a typu vlaku. Vytvorili sme 15 rôznych konfigurácii počtu vrstiev a neurónov na vrstvu. Zo všetkých výsledkov je v tabuľke 20 vybraných niekoľko príkladov. Ako môžeme vidieť, rozširovaním vstupných dát o nové parametre sa presnosť predikcie zlepšila, Najlepšie výsledky sa dosiahli pri použití druhu vlaku, skutočnej aj plánovanej jazdnej doby. V tejto trojici parametrov nesie najvyššiu informáciu skutočná jazdná doba, následne druh vlaku reprezentuje väčšiu pridanú hodnotu ako plánovaná jazdná doba.

V poslednom riadku tabuľky 20 je zhodnotenie najlepšej predikcie a reprezentácia úspešnosti v percentách vzhľadom k skutočnej hodnote a jej odchýlky. Váha, dĺžka aj počet náprav dosahovali podobné hodnoty pri zmene modelu. Zaujímavosťou je počet náprav. Ktorá dosiahla vysokú úspešnosť pri použití len skutočnej jazdnej doby.

EPOCHS	L1	L2	L3	L4	Input	AbsWeight	SqrWeight	AbsLength	SqrLength	AbsAxisC	SqrAxisC	AbsCarC	SqrCar
40	32	64	32	64	T+P+R	181.77	114,397.84	61.54	11,092.10	3.49	40.75	13.15	599.01
40	32	16	32	64	T+P+R	182.57	114,787.22	62.26	11,209.64	3.28	35.42	13.09	584.45
40	16	32	16	32	T+P+R	185.51	115,118.54	62.82	11,352.71	3.35	37.34	13.32	613.66
40	32	16	32	16	T+P+R	185.60	114,789.17	62.96	11,218.31	3.36	37.59	13.30	599.57
40	32	16	16		T+P+R	186.77	117,180.73	63.64	11,399.78	3.41	38.10	13.99	635.86
40	8	16	8	16	T+P+R	187.47	117,649.01	64.57	12,340.37	3.46	39.89	13.90	639.94
40	32	16	32		T+P+R	187.48	119,978.93	64.48	11,616.90	3.39	38.69	13.43	617.51
40	8	16	32	16	T+P+R	187.66	118,152.61	63.48	11,339.29	3.44	39.28	13.86	634.61
40	16	32	64		T+P+R	188.10	119,400.88	64.19	11,661.07	3.39	44.17	13.34	604.15
40	16	32	16		T+P+R	188.62	118,309.17	65.35	11,780.93	3.41	40.04	13.40	647.22
40	16	32	32		T+P+R	188.64	119,919.22	64.78	11,715.81	3.40	38.69	13.48	601.79
40	16	32			T+P+R	190.43	120,156.93	65.75	12,046.49	3.60	41.15	14.12	655.47
40	16	32	16	32	T+R	191.72	118,398.42	66.34	12,432.44	3.66	43.94	14.36	680.00
40	32	16	32	64	T+R	191.76	122,115.72	65.94	12,279.43	3.63	42.90	14.23	677.77
40	32	16	32		T+R	192.06	119,187.98	66.25	12,108.51	3.63	42.93	14.74	777.66
40	32	16	32	16	P+R	283.73	224,865.82	110.03	28,121.18	5.91	97.32	23.80	1,581.03
40	16	32	16	32	P+R	283.97	221,630.94	110.30	29,135.78	5.92	99.58	23.67	1,583.39
40	16	32	64		P+R	284.42	223,498.76	110.77	28,409.59	5.93	96.23	23.80	1,623.79
40	32	16	16		P+R	285.94	226,008.58	111.47	28,698.02	5.99	98.43	23.90	1,631.37
40	16	32	16		P+R	286.12	219,931.05	112.22	28,011.18	5.95	98.08	24.21	1,655.74
40	8	16	32	16	R	305.38	247,282.04	121.49	31,907.42	27.20	1,852.87	6.71	112.73
40	16	32	32		R	306.98	252,175.74	122.67	32,364.69	27.65	1,865.18	6.80	119.51
40	8	16	8	16	R	307.12	249,239.93	121.54	31,561.55	27.36	1,845.21	6.76	113.43
40	32	16	32		R	307.91	250,233.51	121.48	31,799.35	27.15	1,849.13	6.67	111.93
Success rate						65.64%		72.03%		67.20%		84.84%	

** Input - T: TrainType; P: Plan driving time; R: Real driving time*

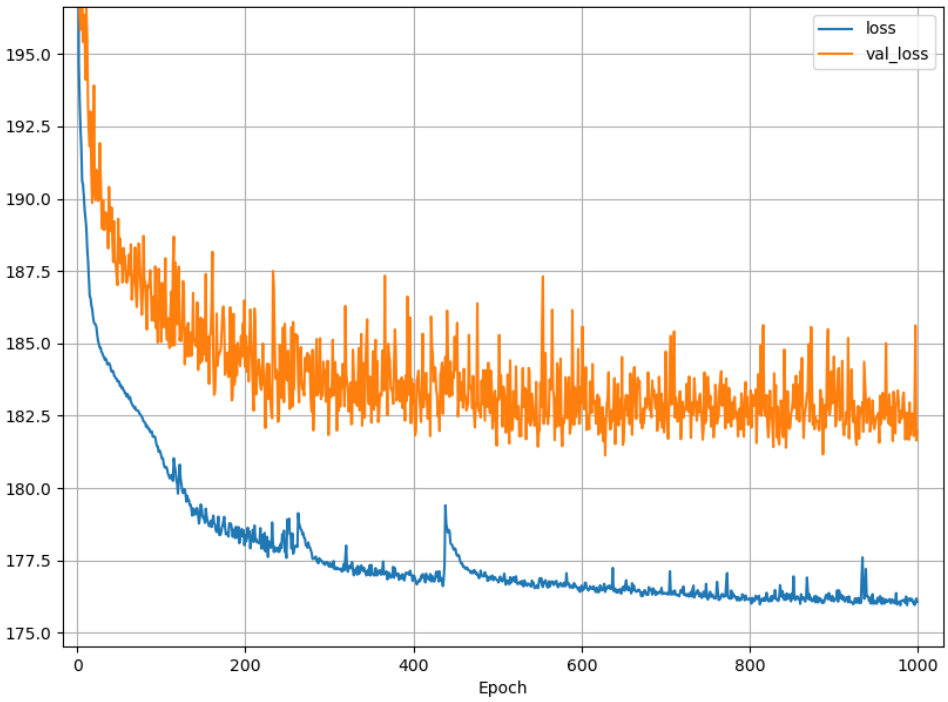
Tabuľka 20 Výber kombinácii vstupných dát, počtu vrstiev a neurónov pri predikcii numerických vlastností vlaku

Ako najlepšiu možnosť predikcie sme zvolili použitie štyroch vrstiev s počtom neurónov postupne 32, 64, 32 a 64. Ako vstupné parametre sme použili všetky tri spomínané a skúsili sme závislosť na zmene parametru počtu opakovaní (Epochs). V tabuľke 21 je zobrazená spolu s presnosťou odhadu aj čas trénovania, ktorý je závislí na počte opakovaní trénovania modelu. Posledný riadok zobrazujúci úspešnosť celkovej predikcie dosahuje pri všetkých parametroch hodnotu okolo 70%.

	Weight			Length			CarCount			AxisCoutn		
epochs	AbsErr1	SqrErr1	Time1	AbsErr2	SqrErr2	Time2	AbsErr3	SqrErr3	Time3	AbsErr4	SqrErr4	Time4
1	249.73	192,961.08	22.84	72.58	20,541.47	33.58	3.66	42.79	32.11	14.98	688.94	34.00
5	191.18	121,211.59	73.92	67.06	11,948.25	75.98	3.48	40.08	80.84	13.86	642.18	90.40
10	187.07	118,547.65	136.05	64.75	11,772.34	138.67	3.45	44.56	131.31	13.68	612.69	126.95
40	183.97	121,630.94	513.86	61.30	11,135.78	421.67	3.41	40.12	421.96	13.67	683.39	464.33
80	183.59	115,423.78	846.70	62.64	11,661.02	677.83	3.31	37.07	660.68	13.32	613.94	697.54
200	180.92	113,607.92	1,808.49	61.65	11,259.31	1,598.58	3.20	35.35	1,406.89	13.07	578.85	1,499.86
500	180.64	111,853.82	4,076.33	61.18	11,240.24	3,693.04	3.19	35.38	3,936.38	12.49	558.96	3,976.63
10000	176.99	111,534.39	7,748.32	60.23	11,044.75	7,922.07	3.22	36.31	9,184.59	12.52	569.93	9,027.55
Succss rate	66.54%			72.62%			67.80%			71.55%		

Tabuľka 21 Zmena parametru epochs pri predikcii numerických vlastností vlakovej súpravy

Vývoj trénovania je zobrazený na obrázku 56. Strata informácie sa výrazne znížila po hodnotu 30, následne zlepšovanie pokračovalo do epochs = 200. Ďalšími opakovanými trénovaniami sa už kvalita predikcie výrazne nezlepšila a tým je to strata výpočtového výkonu pokračovať vo fáze trénovania.



Obrázok 56 Znižovanie straty informácie postupným trénovaním

Záver

V tejto práci sme sa venovali predikcii jazdnej doby vlakovej súpravy v medzistaničnom úseku, možnostiam zvyšovania úspešnosti odhadu a spätnej predikcii vlastností vlakovej súpravy.

Železničná doprava je riadená na základe grafikonu, ktorý pri aktualizácii využíva matematické výpočty k získaniu jazdnej doby v medzistaničnom úseku. Využíva presnú definíciu trate, predpisy a maximálne povolené rýchlosti. V reálnom svete ovplyvňuje jazdnú dobu aj veľa ďalších faktorov. My sme sa sústredili na dosiahnuté jazdné doby v minulosti, ktoré v sebe zahŕňajú všetky vplyvy na jazdnú dobu. Získali sme záznam o všetkých prejazdoch vlakových súprav cez stanice a ich vlastnosti za obdobie päť rokov. Slovenský hydrometeorologický ústav (SHMU) nám poskytol a rozšíril databázu o záznamy o počasí v meste Čadca. Zozbierané dáta sme transformovali do databázy MS Sql pre lepšiu analýzu a prácu s nimi.

K predikcii jazdných dôb sme využili umelú inteligenciu. Zozbierané dáta boli použité pri trénovaní a tým úspešnosť závislá na nich.

Analýzou korektnosti dát sme odhalili anomálie s ktorými sme ďalej nepracovali. Jednalo sa hlavne o neplánované jazdy vlakov, dočasný grafikon počas výluky alebo mimoriadnosti na trati. Reprezentuje to len 0.92% záznamov z celkového počtu 5,264,496 záznamov.

Zlepšenie úspešnosti predikcie závisí na kvalite dát ale aj na vedomosti o zdroji meškaní a spomaľovania vlakovej súpravy. V práci sme analyzovali závislosť skutočnej jazdnej doby na počasí, teplote, zrážkam, sile a smere vetra. Závislosť na type vlaku, jeho dĺžky, váhy, trasy aj času ci obdobiu jazdy. Každé rozdelenie a bližšie špecifikovanie vstupnej množiny zvýšilo presnosť. Avšak selekciou a znížením veľkosti vstupných dát sa znižuje aj spoľahlivosť modelu. Z hľadiska počasia je najväčšie množstvo vlakov vypravovaných pri teplote okolo 5 stupňov, kedy boli zaznamenané aj najpresnejšie jazdy. Hraničné teploty, teda veľké teplo ale aj veľká zima zvyšovali rozptyl jazdných dôb v oboch smeroch. Zaznamenané boli dlhšie jazdné doby a tým zvyšovanie meškania, ale aj skrátenie jazdnej doby. V priebehu dňa je najmenšie množstvo vypravovaných vlakov po polnoci do skorých ranných hodín a meškania vznikajú najviac v poslednej štvrtine dňa medzi 18 a 24 hodinou. V prípade ročných období sme zistili, že najväčšie meškania sú v jesennom období v mesiacoch September, Október a November. A počet

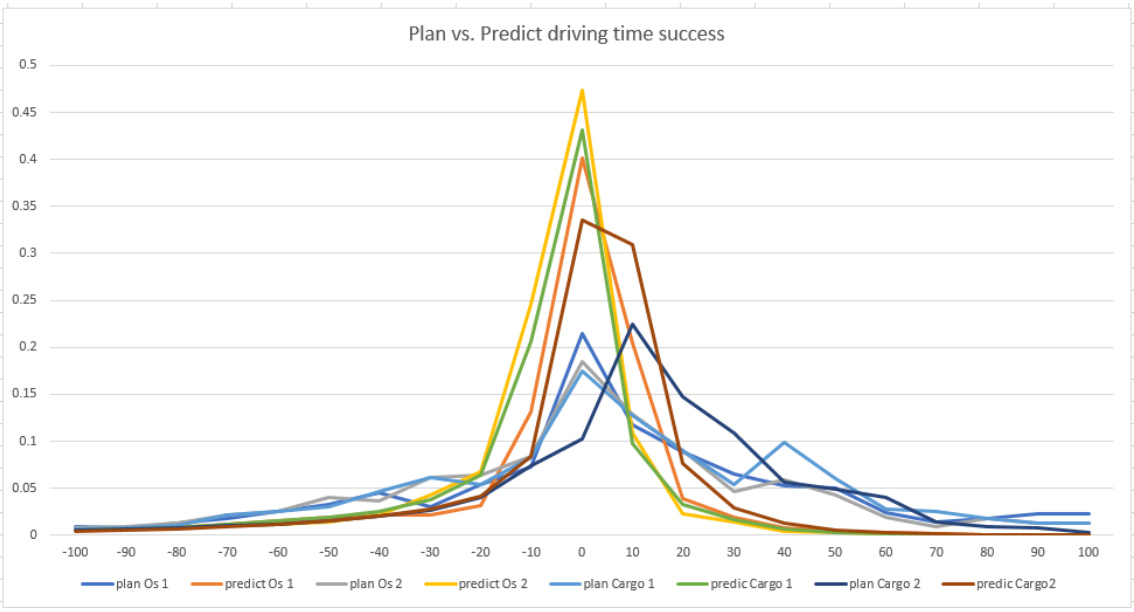
vlakov v priebehu roka je uniformne rozložený. Špecifické správanie a nerovnomerný pomer jednotlivých typov vlakov nám priniesol najúspešnejšie predikcie. V prípade osobných vlakov sme sa dostali na úspešnosť 94,17% a celkovo zlepšili predikciu o 5% v porovnaní s plánovanou jazdnou dobou. Medzistaničných úsekoch v našej skúmanej oblasti medzi Přerov a Ostravou je 22, každý úsek má iné špecifikácie, sklon, polomer zákrut a technické vybavenie. Taktiež záleží v ktorom smere je jazdná doba predikovaná a vzniká nám 44 možností rozdelenia. Pre prehľadnosť sme náhodne vybrali 8 reprezentatívnych vzoriek s hustou ale aj menej hustou dopravou. Skutočná jazdná doba sa líšila od plánovanej v priemere o 10,58% čo sa nám podarilo znížiť na 7,17% a dosiahnuť úspešnosť 92,83%.

Vytváranie modelov sme programovali v jazyku python, C# a na porovnanie použili implementáciu klaudových riešení. V prípade klaudov boli možnosti implementácie obmedzené, ale ponúkali výhodu vo veľkom výpočtovom výkone počas trénovania bez potreby vlastniť výkonný hardvér. Následne natrénovaný model už nepotrebuje výkonný server a môže bežať na prenajatých serveroch za zlomkovú cenu. V prípade lokálneho vývoja v python a C# sme nepresiahli 6 hodín trénovania. V jazyku Python sme implementovali algoritmus Boosted trees a konvolučné neurónové siete. Oba algoritmy sú upravovateľné niekoľkými parametrami. Nami najlepšie nastavenie dosiahlo úspešnosť 90,9% v prípade Boosted Trees a 92,5% v prípade konvolučných sieti pri použití 4 vrstiev s 64 uzlami na vrstvu. V programovacom jazyku C# sme použili algoritmy FastTree, SDCA a FastForest. Najlepšie z nich dopadol algoritmus FastTree ktorý dosiahol úspešnosť 91,29%. Porovnanie algoritmov prebiehalo nad všetkými dátami a použitím selekcie sa výsledná predikcia ešte zvýšila.

Spomenuté predikcie boli vypočítané na základe parametrov a vlastností vlakovej súpravy. Jedná sa o off-line predikciu, kde nevstupujú do modelu aktuálne namerané hodnoty. Kombináciou selekcie a algoritmu umelej inteligencie. V prípade selekcie sme využili rozdelenie podľa medzistaničného úseku a na základe typu vlaku do troch skupín osobných, nákladných a ostatných vlakov. A algoritmu konvolučných neurónových sieti sme dosiahli najlepšiu off-line predikciu na hodnote 93,73% priemerne na všetkých dátach. Selekcia rozdelila dáta tak, že každý medzistaničný úsek obsahuje tri modely podľa typu vlaku. Výsledná aplikácia určí trať a typ vlaku, podľa čoho vyberie konkrétne natrénovaný model.

Vedomosti o aktuálnom stave jazdnej súpravy sme využili v ďalšej časti. Aktuálne problémy, opotrebenie a iné faktory môžu dočasne ovplyvňovať schopnosti súpravy dosahovať plánované jazdné doby, ktoré sa nedajú predikovať len z čísla vlaku, či jeho typu. Vybrali sme si jeden medzistaničný úsek a sledovali dosiahnuté jazdné doby a státie v staniciach predchádzajúcich úsekov. Nevýhodou tohto riešenia je nevyhnutnosť znalosti jazdných dôb z predchádzajúcich medzistaničných úsekoch čo môže byť problém hlavne na okrajových úsekoch riadenej oblasti. Museli by sme túto informáciu získavať zo susednej oblasti pri príchode jazdnej súpravy do našej riadenej oblasti. Keďže predpovedáme jazdnú dobu na základe aktuálne nameraných hodnôt, voláme to tiež predikcia v reálnom čase. Úspešnosť modelu 90,37% je o niečo nižšia ako v prípade off-line predikcie avšak našim cieľom je ich kombinácia.

Na obrázku 57 sú zobrazené výsledky výsledného programu, ktorý kombinuje on-line a off-line predikciu. Pre prehľadnosť sú vybrané 2 trate a predikcia prebehla nad osobnými a nákladnými vlakmi. Ideálnym grafom by bol Dirakov impulz, ktorý by nadobúdal hodnotu „1“ v chybe „0“ reprezentujúca 100% záznamov s nulovou chybou a pre zvyšné hodnoty by bola hodnota „0“. My môžeme vidieť, že grafy predikcii sú užšie a vyššie. To znamená že viac záznamov bolo predikovaných s menšou chybou.



Obrázok 57 Predikcia výsledného modelu

K spájaniu sme vyskúšali tri algoritmy, kde sa potvrdil ako najlepší výpočet „Highest diff“ podľa rovnice:

=IF(ABS([@pred1]-[@plan])>ABS([@pred2]-[@plan]),[@pred1],[@pred2])

Ktorá vyberala predikciu s väčším absolútnym rozdielom od plánovanej jazdnej doby. Týmto spôsobom sme využili off-line predikciu s úspešnosťou 93.73% a on-line predikciu s úspešnosťou 90,37% a dosiahli výslednú hodnotu 94,38%. Čo znamená, že nami predikovanej jazdná doba sa líši len o 5,62% od skutočnej jazdnej doby.

Skutočné jazdné doby sa odlišujú od plánovaných o 85,61%, našu pridanú hodnotu dizertačnej práce a presnejší odhad o 8,77% jazdnej doby by sme chceli využiť aj v praxi. K tomuto účelu sme implementovali internetovú službu v jazyku C#, ktorá očakáva na vstupe parametre jazdnej súpravy. Podporuje použitie off-line predikcie, len na základe vlastností jazdnej súpravy, on-line predikciu a aj ich kombinácie v prípade, že sú vyplnené všetky dostupné informácie o vlakovej súprave.

Záverečnou časťou sme sa venovali spätnej predikcii parametrov vlakovej súpravy. Implementácia sa dá využiť pri kontrole vstupnej trénovanej množiny, prípadne ako predspracovanie záznamu k predikcii k zlepšenie úspešnosti samotnej predikcie. Začali sme odhadom typu vlaku na základe skutočnej jazdnej doby. Použili sme konvolučné neurónové siete zamerané na kategorickú premennú. Nastavenie s najlepšiu úspešnosťou dosiahli tri vrstvi s hodnotami neurónov na jednotlivých vrstvách 16, 32 a 64. Keďže vstupný parameter bol len jeden a výstupných kategórii 10, dosiahli sme úspešnosť 55,01%. Rozšírením vstupnej hodnoty skutočnej jazdnej doby o plánovanú jazdnú dobu a váhu vlakovej sústavy sa nám podarilo zvýšiť úspešnosť prvej predikcie na 86,63%. V prípade, že by sme akceptovali dva najpravdepodobnejšie typy vlaku, dostali by sme sa na úspešnosť 94,43%. Najčastejšie vyskytujúcou chybou odhadu bol vlak typu *Priebežný nákladný* označený ako *Nákladný expres* a *Rýchlik* označený ako *Expresný* vlak.

Medzi numerické parametre vlaku patria váha, dĺžka, počet náprav a vagónov. Z poznatkov o predikcii typu vlaku sme vytvorili niekoľko kombinácii parametrov neurónových sieti a aplikovali na všetky kombinácie vstupných parametrov zo skupiny typ vlaku, skutočná a plánovaná jazdná doba. Dosiahnuté výsledky sú v tabuľke 20. Najlepšiu predikciu sme dosiahli v prípade počtu vagónov na hodnote 84,84%, váhy vlaku na 65,64% a dĺžku vlakovej súpravy na 72,03%.

Riadenie železničnej dopravy v prípade bezporuchového stavu nie je problém a vlaky by dodržiavali plánované jazdné doby. Aj v nich je už zahrnutá rezerva a tiež

možnosť optimalizácie spotreby elektrickej energie šetrným využívaním akcelerácie a brzdenia. To vedia využiť vo svoj prospech v prípade, že jazdnej súprave vzniklo meškanie pomocou menej ekonomickej jazdy. Našou prácou sme sa pokúsili predpovedať vznik komplikácii podľa historicky opakovaných sa udalostí. Podarilo sa nám zvýšiť úspešnosť predikcie jazdnej doby a tým využiť program pri riadení železničnej dopravy upozornením dispečera o predpoklade vzniku meškania pre daný spoj. Ďalším využitím je integrácia do systému ASVC, ktorý zabezpečuje automatické stavanie ciest. Program na výpočet predikcie jazdnej doby pomocou umelej inteligencie je dostupný ako online služba a tým je zjednodušená jeho integrácia a využitie.

Citácie

[1] J. RALPH, „Improving arrival time prediction of Thailand's passenger trains using historical travel times,“ 2013. [Online]. Available: https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-cpg-retail-manage-it. [Cit. 14 4 2019].

[2] V.-T. Ostrava, „Lokomotivní simulátor,“ [Online]. Available: http://www.342.vsb.cz/loksim/FRVS_Simul%C3%A1tor_3-4.pdf.

[3] S. &. P. T. &. K. N. &. C. K. Pongnumkul, „Improving arrival time prediction of Thailand's passenger trains using historical travel times,“ 2014. [Online]. Available: https://www.researchgate.net/publication/282208077_Improving_arrival_time_prediction_of_Thailand's_passenger_trains_using_historical_travel_times.

[4] &. Q.-p. Z. Pu W., „Train delay analysis and prediction based on big data fusion, Transportation Safety and Environment,“ 2019. [Online]. Available: https://academic.oup.com/tse/advance-article/doi/10.1093/tse/tdy001/5306170.

[5] „A history of machine learning. in Data & Analytics,“ [Online]. Available: https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning.

[6] K. MARDEN, „Top 10 real-life examples pf Machine Learning in Machine Learning,“ 2018. [Online]. Available: https://bigdata-madesimple.com/top-10-real-life-examples-of-machine-learning.

[7] K. ERIK, „Úvod do umelej inteligencie in Inteligentné mechatronické systémy | Prednáška 3,“ [Online]. Available: http://uamt.fei.stuba.sk/MVI/sites/default/files/Prednaska%203%20uvod%20do%20UI.pdf.

[8] T. Gupta, „Deep Learning:Feedforward Neural Network,“ 5 January 2017. [Online]. Available: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7.

[9] L. Loffler a T. Kello, „Využitie dát o jazdných dobách vlakovej súpravy na odhad jej parametrov a jazdných dôb na zvyšných úsekoch trate,“ University of Zilina, Zilina, 2020.

[10] E. Kostelansky a T. Kello, „Využitie frameworku strojového učenia ML.NET k analýze vlastností vlakovej súpravy vzhľadom na jej jazdnú dobu,“ University of Zilina, Zilina, 2019.

[11] R. Dalinina, „Introduction to correlation,“ 2017. [Online]. Available: https://www.datascience.com/blog/introduction-to-correlation-learn-data-science-tutorials.

[12] D. Hanculak a T. Kello, „Implementácia strojového učenia pomocou frameworku Tensorflow,“ University of Zilina, Zilina, 2019.

[13] V. d. Berg, „Cramér’s V - What and Why?,“ [Online]. Available: https://www.spss-tutorials.com/cramers-v-what-and-why.

[14] F. Ondrusek a T. Kello, „Porovnanie klaudových systémov strojového učenia k analýze vlastností vlakovej súpravy,“ University of Zilina, Zilina, 2019.

[15] P. CHRISTIAN, „Understanding Regression Error Metrics in Python,“ 2018. [Online]. Available: https://www.dataquest.io/blog/understanding-regression-error-metrics.

[16] P. N., Machine Learning with Spark, Livery Place: Packt Publishing Ltd ISBN 9781783288519, 2015.

[17] S. Yegulalp, „What is TensorFlow? The machine learning library explained,“ 2018. [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

[18] B. Winkler, „Python overtakes R, becomes the leader in Data Science,“ 2017. [Online]. Available: https://www.kdnuggets.com/2017/08/python-overtakes-r-leader-analytics-data-science.html.

[19] „Co je cloud computing?,“ [Online]. Available: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/.

[20] Google, „Products and services,“ 2019. [Online]. Available: https://cloud.google.com/products/.

[21] R. MARGARET, „Microsoft Azure (Windows Azure,“ 2018. [Online]. Available: https://searchcloudcomputing.techtarget.com/definition/Windows-Azure.

[22] J. VANDERPLAS, „Python Data Science Handbook.,“ [Online]. Available: https://jakevdp.github.io/PythonDataScienceHandbook.

[23] P. Ing. Pavel Krýže, 31 10 2018. [Online]. Available: https://provoz.spravazeleznic.cz/portal/Show.aspx?path=/Data/Mapy/rychlosti.pdf

Zoznam vlastných publikácií

Kello Tomáš, Kršák Emil „Methods to estimate train arrival time“, Mathematics in science and technologies 2019, s. 45-50, ISBN: 978-1-7940-0218-0

Kello Tomáš, Kršák Emil „Analyze train data set with use of linear regression and cloud systems“, International Conference on Information and Digital Technologies 2019, s. 203-211, ISBN: 978-1-7281-1401-9

Kostelánsky Erik, Kršák Emil, Kello Tomáš „Estimate Train Driving Time with Artifical Inteligence“, International Conference on Information and Digital Technologies 2019, s. 237-244, ISBN: 978-1-7281-1401-9

Jana Dudova, Emil Krsak, Tomas Kello „Pre-diagnostic of diabetes with the use of statistical machine learning“, IEEE 15th International Scientific Conference on Informatics 2019, s. 433-438, ISBN: 978-1-7281-3178-8

Emil Krsak, Tomas Kello „Improving teoretic train driving time with AI and TensorFlow“, ISIA 2020

Emil Krsak, Tomas Kello „Improving learning step in artificial intelligence with categorization“ 18th IEEE International conference on emerging elearning technologies and applications 2020, s. 433-438, ISBN 978-1-7281-3178-8.