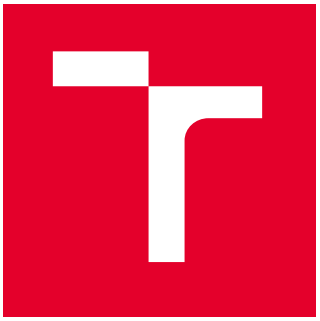


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KLASIFIKACE VOZIDEL NA ZÁKLADĚ POČTU A VZDÁLENOSTÍ NÁPRAV

CLASSIFICATION OF VEHICLES BASED ON THE NUMBER AND SPACING OF AXLES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Milan Bašo**

AUTHOR

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

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Milan Bašo ***ID:*** 230037

Ročník: 3 ***Akademický rok:*** 2024/25

NÁZEV TÉMATU:

Klasifikace vozidel na základě počtu a vzdáleností náprav

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout klasifikátor vozidel pracující na základě dat ze dvou nápravových senzorů (piezoelektrické pásky nainstalované v celé šíři jízdního pruhu) s pevným rozestupem (1-5 m). Návrh musí zohledňovat typické problematické dopravní scénáře a vlivy (široký rozsah rychlostí a jejich změny, dopravní zácpy, různé směry jízdy, změny jízdních pruhů apod.) i omezení výpočetních prostředků.

1. Seznamte se s danou problematikou.
2. Vhodně upravte, roztřiďte, případně anotujte dodaný vzorek dat.
3. Navrhněte a implementujte algoritmy pro separaci vozidel - modul, který je schopen označovat skupiny odezev z dvou řad časových posloupností, které patří jednomu vozidlu.
4. Navrhněte klasifikátor vozidel. Zvažte různá klasifikační schémata (analýza, kolik tříd je možno rozlišit s rozumnou přesností) a různé varianty vstupů (s/bez předzpracované rychlosti, s/bez plochy/maxima odezvy). Zvažte sloučení separace a klasifikace do jednoho kroku.
5. Vybrané postupy klasifikace implementujte.
6. Vše řádně otestujte a zhodnoťte. Definujte omezující podmínky.

DOPORUČENÁ LITERATURA:

González, B.; Jiménez, F.J.; De Frutos, J. A Virtual Instrument for Road Vehicle Classification Based on Piezoelectric Transducers. Sensors 2020, 20, 4597. https://doi.org/10.3390/s20164597

Termín zadání: 10.2.2025 ***Termín odevzdání:*** 28.5.2025

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

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

ABSTRAKT

Bakalárska práca sa zaoberá zhlukovaním a klasifikáciou vozidiel využitím piezoelektrických senzorov uložených na vozovke. V práci sú popísané jednotlivé algoritmy zhlukovania a klasifikácie, kde sú následne zhodnotené ich výhody a nevýhody a ich účinnosť.

KĽÚČOVÉ SLOVÁ

klasifikácia, piezoelektrické senzory, oddelenie, python, náprava, vozidlo, zhlukovanie, perceptron, neurónová sieť

ABSTRACT

The thesis focuses on the clustering and classification of vehicles using piezoelectric sensors installed on roads. Various methods for clustering and classification are analyzed and evaluated based on their performance.

KEYWORDS

classification, piezoelectric sensors, separation, python, axle, vehicle, clusterisation, perceptron, neural network

Vysadené pomocou balíčku `thesis` verzie 4.11; `https://latex.fekt.vut.cz/`

BAŠO, Milan. *Klasifikace vozidel na základě počtu a vzdáleností náprav*. Bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2025. Vedúci práce: Ing. Ilona Janáková, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Milan Bašo
VUT ID autora:	230037
Typ práce:	Bakalárska práca
Akademický rok:	2024/25
Téma závěrečnej práce:	Klasifikace vozidel na základě počtu a vzdáleností náprav

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

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

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som sa poďakoval vedúcej bakalárskej práce pani Ing. Ilone Janákovej, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Úvod

Klasifikácia vozidiel v modernej doprave je veľmi dôležitý prvok na analýzu osobnej alebo cargo dopravy. Najmä ťažké nákladné vozidlá vplývajú na výdrž cestnej infraštruktúry. Získaním dát o zložení dopravy je možné eliminovať napr. nekvalitný asfalt a prispôsobiť ho dopravnej záťaži. Nemenej dôležitou súčasťou je aj klasifikácia vozidiel na účely vyberania mýtnych poplatkov.

Cieľom práce je zanalyzovať, ako je potrebné upraviť alebo anotovať dáta, oddeliť vozidlá od seba a aký klasifikačný algoritmus je vhodný na tento typ dát. Práca sa zameriava na piezoelektrické senzory umiestnené na vozovke v celej šírke jazdného pruhu.

V kapitole 1 sú spracované spôsoby zberu dát, princíp fungovania piezoelektrických snímačov a výpočty parametrov zo snímačov. V druhej kapitole sa práca venuje teoretickému rozboru ohľadom metód strojového učenia. V tretej časti je práca zameraná na softvér.

1 Úvod do klasifikácie vozidiel

Vozidlá je možné klasifikovať podľa niekoľkých možných klasifikačných schém. Konkrétne klasifikátory používajú rôzne charakteristiky, podľa ktorých zaradujú vozidlá do kategórií. Najznámejšie klasifikačné schémy využívajú nasledovné vlastnosti:

- počet a veľkosti rázvorov náprav,
- dĺžka vozidla,
- pohon alebo využívané palivo,
- hmotnosť
- typ karosérie alebo pripojeného vozidla.

Dôležitým faktorom pri klasifikácii vozidiel je použitie vhodnej technológie na zber dát. Väčšina súčasných technológií dokáže zbierať len konkrétne údaje, resp. vlastnosti a preto je dôležité vopred určiť, ktoré dáta chceme zbierať a vyhodnocovať. Môže nastať situácia, kde sa bude musieť použiť špecifická technológia vzhľadom na vlastnosti vozovky alebo iného problému (nemožnosť rezať do asfaltu,...). Konkrétnym prípadom je použitie indukčných slučiek zapojených v sérii. Touto realizáciou sa určí dĺžka vozidla, ale nemožno určiť rázvory náprav podľa niektorej klasifikačnej schémy bez dodatočnej elektroniky.

1.1 Metódy zberu dát

1.1.1 Krátkodobý zber dát

V minulosti bol využívaný na zber dát človek, ktorý bol pri vozovke a manuálne sčítaval vozidlá, ich počet náprav a ďalšie požadované vlastnosti. Metóda je ale ovplyvnená ľudským faktorom a v súčasnosti je aj prevádzka tohto typu zberu drahšia ako technológie. Preto sa zaviedol automatizovaný zber dát. Tento typ zberu používa senzory položené na vrchu vozovky. Najčastejším je zber rázvorov náprav alebo celkovej dĺžky vozidiel.

Firmy podnikajúce v tomto odvetví vytvárajú kompletne prenositeľné riešenia. Zber dát musí byť s čo najväčšou presnosťou, riešenie je možné ľubovoľne presúvať z jednej lokality do druhej, musí mať vhodne navrhnuté napájanie vzhľadom na nedostupnosť konvenčných zdrojov elektrickej energie a cena musí byť stlačená nadol.

Najčastejším časovým úsekom na zber dát pomocou prenositeľného riešenia je 24 alebo 48 hodín.

Medzi najpoužívanejšie riešenia patria[3]:

- trubičkový cestný senzor,
- piezoelektrický senzor,
- optické vlákno,

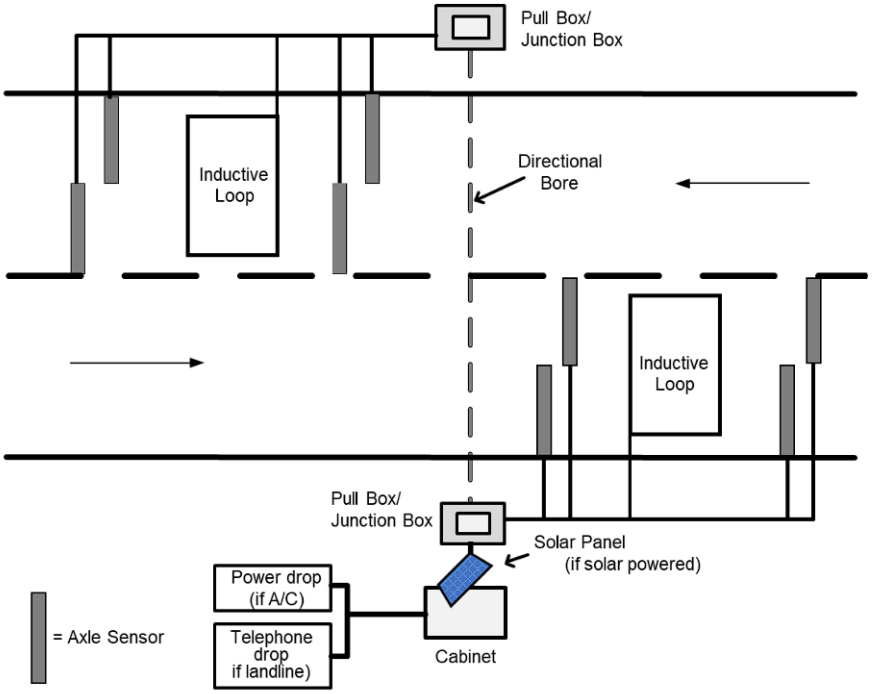
- prenositeľné indukčné slučky,
- magnetometer.

Prvé tri technológie poskytujú dostatočný počet dát na klasifikáciu vozidiel a rozdelenie do skupín podľa príslušnej klasifikačnej schémy. Tieto technológie nie sú cenovo náročné a majú jednoduchú štruktúru na umiestnenie na vozovku. K nevýhodám patrí nemožnosť rozlíšiť dve vozidlá v tesnom vedení za sebou. Nevhodné je aj umiestnenie senzorov na miesta, kde sa vyskytujú dopravné zápchy. Tieto senzory vo väčšine prípadov berú dve vozidlá, ktoré idú za sebou ako jedno väčšie vozidlo s viacerými nápravami.

Na prvý pohľad je metóda krátkodobého zberu dát kompaktná na umiestnenie, no je nutné odklonenie dopravy z miesta inštalácie. To už môže ovplyvniť cenovú alebo aj časovú efektívnosť.

Jednou z možností, ako zbierať dáta, je aj permanentné umiestnenie senzorov do vozovky a následný zber vykonávať vo zvolených časových úsekoch. Miesto inštalácie v danom prípade nemusí mať pripojenú elektroniku na vyhodnocovanie.

Na obrázku č.1.1 je znázornená kombinácia dvoch druhov senzorov a pripojenie vyhodnocovacej elektroniky. Signály zo senzorov sú konvertované a vyhodnocované elektronickým kontrolérom (zvyčajne komerčne dostupný embedded systém) ako počty a typy vozidiel. Dáta sú uchovávané na úložisku on-demand typu cloudové úložisko.



Obr. 1.1: Príklad uloženia kombinácii senzorov na vozovku[1]

1.1.2 Kontinuálny zber dát

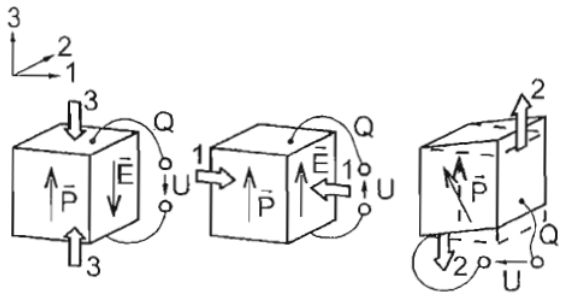
Umiestnenie senzorov na vozovku postačuje na krátkodobý zber dát, ale pre kontinuálnu klasifikáciu je tento spôsob nevhodný najmä kvôli opotrebovaniu úchytov senzorov. Na senzory vo vozovke nepôsobí ani narazenie kolesa pri vstupe. Ďalším obmedzením je aj prívod napájania, kým prenosné technológie si môžu vystačiť s akumulátorom, dlhodobejšie merania už musia mať zabezpečený prívod elektriny alebo si ju vytvárať. Nevýhodou je dlhšie odstavenie dopravy na mieste inštalácie z dôvodu zásahu do vozovky. Následne ale už nie je potrebná špeciálna údržba, len periodické prehliadky. Pri umiestňovaní senzorov do vozovky je treba brať ohľad aj na možnosť rekonštrukcie vozovky alebo rozširovanie či zmena pruhov.

Nesprávnym postupom pri inštalácii senzorov do vozovky sa zvyšuje riziko poškodenia. Riziko poškodenia hrozí aj pri zásahu blesku do bodu blízkeho senzorom alebo aj zatečenie vody. Nekvalitná vozovka môže spôsobiť aj započítanie tzv."ghost"náprav. Ak náprava vozidla poskakuje kvôli nerovnostiam a je blízko meracej stanice, je pravdepodobné, že vozovka deformuje senzor a ten to vyhodnotí ako impulz od prechodu nápravy. [1][3]

1.2 Piezoelektrické senzory

Fyzikálnou podstatou týchto senzorov je piezoelektrický jav založený na polarizácii polokryštalických alebo kryštalických dielektrík vtedy, ak sú vystavené mechanickému napätiu (priamy piezoelektrický jav) alebo v deformácii kryštálov pri pôsobení vonkajšieho elektrického poľa (nepriamy piezoelektrický jav).

V senzoroch, pri priamom piezoelektrickom jave, pôsobí mechanické napätie buď kolmo na elektródy pre zber náboja (pozdĺžny jav), rovnobežne s ich rovinou (priečny jav) alebo často využívaná metóda, šmyková deformácia. Jej výhoda spočíva vo väčšej citlivosti a menších rušivých účinkoch teplotných dilatácií konštrukcie senzoru. Obrázok č.1.2 zobrazuje jednotlivé deformácie.



Obr. 1.2: Pozdĺžny, priečny a strihový piezoelektrický jav[2]

Typické materiály pre piezoelektrické senzory sú z týchto skupín[2]:

- Monokryštály, napr. triglycinsulfát, oxid kremičitý.
- Polykryštalické keramické materiály, napr. titaničitan bárnatý.
- Organické polyméry, napr. polyvinylidendifluorid (PVDF).

1.3 Použitie tejto technológie v klasifikácii

Piezo efekt v senzore je dynamický. Elektrický náboj zo senzora je vyvolaný zmenou sily pôsobiacej naň. Pri dopravnej zápche alebo ak sa vozidlá budú pohybovať pomalou rýchlosťou pod určitou hranicou, je tento druh senzoru nepoužiteľný.
Piezokeramický senzor má veľkosť približne ako komerčný koaxiálny kábel. Pri použití vo WIM alebo klasifikácii je typicky uložený v hliníkovom kanáli a zaliaty epoxidom. Kanál je potom uložený v jednej rovine s vozovkou.

Problémom týchto senzorov je dodržanie uniformity v celej dĺžke senzoru. Senzory, ktoré sa využívajú na klasifikáciu, ale nemusia spĺňať také prísne kritériá ako senzory na WIM(weight in motion) technológiu.
Piezopolymérový senzor má podobné vlastnosti ako piezokeramický senzor. Rozdielom je uloženie čidla do mosadzného púzdra.
Piezokremeňový senzor je unikátny v malej citlivosti na zmeny teplôt. Nevýhodou je vyššia obstarávacia cena.

Spomenuté druhy piezoelektrických senzorov majú pri použití v klasifikácii podobné rozloženie na vozovke, ale trochu iné operačné podmienky a iné požiadavky na inštaláciu.

Minimom pre klasifikáciu je použitie aspoň dvoch paralelne umiestnených senzorov. Možným rozložením je aj pridanie indukčnej slučky (zvyčajne medzi senzory) a tým zlepšiť oddelenie vozidiel spôsobom informovania kontroléru, že koniec prvého vozidla prešiel slučkou. Ďalšími možnými rozloženiami senzorov sú:

- dve indukčné slučky a medzi nimi jeden piezo snímač,
- dve indukčné slučky a dva piezo snímače.

Piezoelektrické senzory môžu byť opakovane zaliate asfaltom a ich funkčnosť nebude ovplyvnená.[3]

1.4 Výpočty parametrov zo senzorov

1.4.1 Rýchlosť

Prakticky každá technológia používaná v metrológii dopravy poskytuje aspoň teoretický výpočet rýchlosti pohybujúceho sa vozidla. Nemusí však plniť dôležitú úlohu

pri vyhodnocovaní dopravy. Pri technológiách na klasifikáciu vozidiel je tento parameter jedným z oporných bodov správnej klasifikácie.

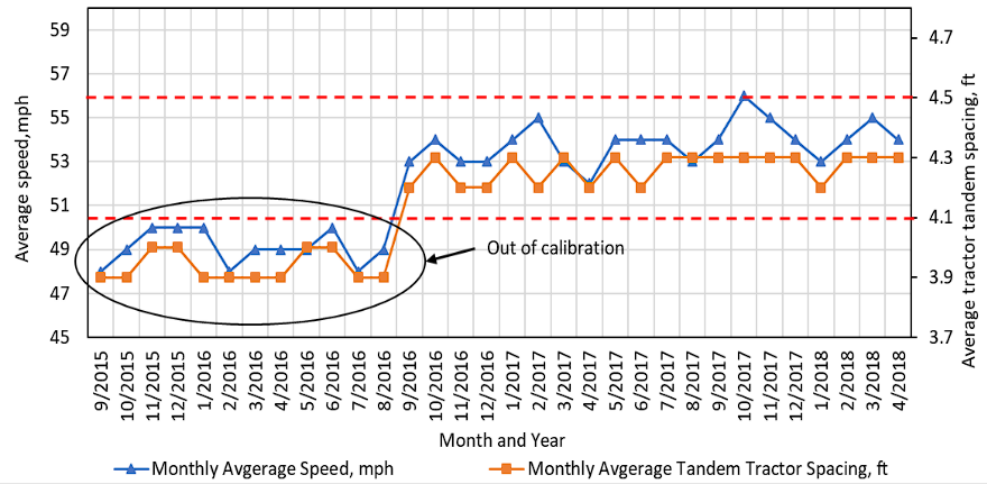
Aplikácie s jednou indukčnou slučkou využijú tvar signálu a aproximujú rýchlosť.

Pre nás najpodstatnejším typom merania rýchlosti je umiestnenie dvoch senzorov paralelne s pevnou vzdialenosťou medzi sebou. Tento typ radenia senzorov vytvorí rýchlostnú pascu.

Pri správnej kalibrácii oboch senzorov je už jednoduché použiť výpočet rýchlosti ako dráhu za určitý čas. V rovnici č. 1.1 je na výpočet použitá vzdialenosť medzi senzormi d_{12} vydelená rozdielom časov T_1 a T_2 vstupov nápravy na oba senzory.

$$v = \frac{d_{12}}{T_2 - T_1} \tag{1.1}$$

Na obrázku 1.3 sú zobrazené priemerné mesačné rýchlosti vozidla. V čiernom kruhu sú zakrúžkované priemerné rýchlosti, ktoré naznačujú, že senzory nie sú správne nakalibrované.



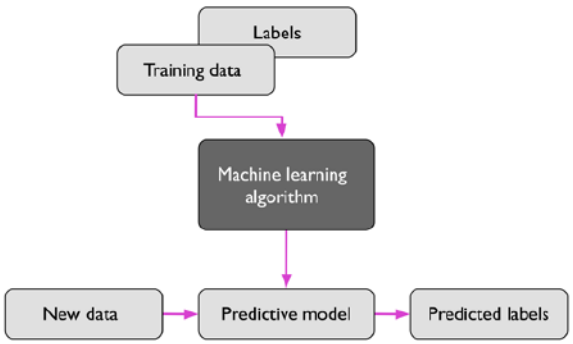
Obr. 1.3: Rozkalibrované senzory[1]

2 Strojové učenie

2.1 Strojové učenie s učiteľom

Cieľom v strojovom učení s učiteľom je naučiť navrhovaný model **označenými** tréningovými dátami. Tento postup nám umožňuje robiť predikcie výsledkov z neznámych dát. Učiteľ v tejto metodike znamená, že vstupné dáta (tréningové vzory) majú už známy výstupný signál. Učenie s učiteľom modeluje vzťah medzi vstupom a výstupnými dátami (označené dáta).

Na obrázku 2.1 sú zobrazené označené (label) dáta, ktoré sú spracované algoritmom (machine learning algorithm). Pomocou algoritmu sa nastavia dáta do prediktívneho modelu, ktorý dokáže vytvárať predikcie na neoznačené vstupné dáta (new data). [4]



Obr. 2.1: Grafické znázornenie algoritmu pomocou učenia s učiteľom[4]

2.1.1 Logistická regresia

Logistická regresia je regresný algoritmus, ktorý môže byť použitý na klasifikáciu. Algoritmus vypočíta pravdepodobnosť aktuálne riešeného prípadu. Ak je pravdepodobnosť vyššia ako zadaná hranica, model vyhodnotí danú inštanciu ako pozitívnu (1). Ak je pravdepodobnosť nižšia, inštancia je negatívna alebo nula. Model sa správa ako binárny klasifikátor.[6]

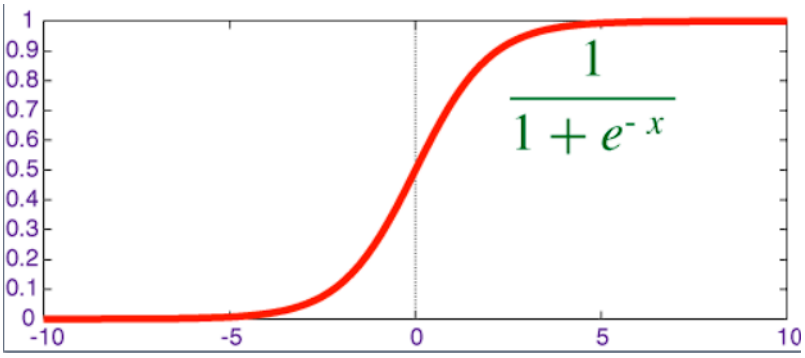
Klasifikácia je kvalitatívne vyjadrenie výstupnej premennej. To znamená, že nepredikujeme hodnotu, ale kategoricky zvolenú premennú priradíme k vstupnej premennej. [4]

Podobne ako lineárna regresia, logistická regresia vypočíta váženú sumu všetkých vstupných vlastností (features). Výsledok je potrebné ešte spracovať.[6]

Podstatnou časťou je výber vhodnej funkcie, ktorá vie rozprestrieť výstupy modelu od 0 po 1 pre všetky vstupné hodnoty X. Rovnica č.2.1 je používaná pre logistickú regresiu.β_0 a β_1 sa nazývajú regresné koeficienty známe z lineárnej regresie. β_0 sa nazýva aj úrovňový koeficient. Predstavuje priesečník s osou y. β_1 je smernica regresnej priamky. Ak je koeficient kladný, tak so stúpajúcou hodnotou premennej X je aj závislá premenná Y rastúca. V prípade, že je koeficient záporný, s rastúcou hodnotou premennej X je závislá premenná Y klesajúca. Rozdielna je však zmena závislej premennej Y prepočítaná na jednu jednotku zmeny vstupnej premennej X. Tento jav je daný nelineárnym vzťahom medzi vstupnou premennou X a závislou premennou Y.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{2.1}$$

Po úprave na tvar rovnice č.2.2 nám vznikla funkcia, ktorá sa nazýva logit. Funkcia mapuje pravdepodobnosť [0,1] do oboru reálnych hodnôt. Výpočtovo je ale výhodnejšia inverzná funkcia sigmoid k funkcii logit. Táto funkcia mapuje obor reálnych hodnôt do oboru pravdepodobnosti [0,1]. [5][7]



Obr. 2.2: Sigmoid funkcia[9]

$$\log\left(\frac{p(X)}{p(X)-1}\right) = \beta_0 + \beta_1 X \tag{2.2}$$

Výpočet koeficientov β_0 a β_1 je možný pomocou nelineárnej metódy najmenších štvorcov. Efektívnejšou alternatívou je výpočet pomocou metódy maximálnej dôveryhodnosti.

Metóda maximálnej dôveryhodnosti odhaduje hodnoty parametrov β_0 a β_1 do modelu p(X) z tréningových dát (X). Maximalizáciou funkcie dôveryhodnosti dosiahneme, že tréningové dáta majú najväčšiu pravdepodobnosť. Vyjadruje pre rôzne hodnoty parametrov β_0 a β_1 pravdepodobnosť, ktorou budú dáta klasifikované do jednej z dvoch tried (napr. 1 alebo 0).

V rovnici č.2.3 je vyjadrená funkcia dôveryhodnosti. Súčinom súčinov maximalizovaných pravdepodobností pre obe triedy odhadneme parametre β_0 a β_1.[5][8]

$$l(\beta_0, \beta_1) = \prod_{i:y=1} p(x_i) \prod_{i:y=0} (1 - p(x_i)) \tag{2.3}$$

Logistická regresia s viacerými nezávislými premennými X je použitie viacerých premenných X na predpoveď závislej premennej Y. V rovnici č.2.4 je zmena oproti rovnici č.2.2 len v množstve nezávislých premenných X.

$$\log\left(\frac{p(X)}{p(X) - 1}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{2.4}$$

Logistická regresia s viacerými triedami je klasifikátor s počtom tried väčším ako 2. Rovnica č.2.5 je rozdielna oproti rovnici č.2.1 v pridanom súčte v menovateli.

$$Pr(Y = k|X = x) = \frac{e^{\beta_{k0} + \beta_{k1} x_1 + ... + \beta_{kp} x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_{l1} x_1 + ... + \beta_{lp} x_p}} \tag{2.5}$$

Po úprave na tvar prirodzeného logaritmu je vidieť linearitu závislých premenných Y medzi všetkými pármi klasifikačných tried.[5]

$$\log\left(\frac{Pr(Y = k|X = x)}{Pr(Y = K|X = x)}\right) = \beta_{k0} + \beta_{k1} x_1 + ... + \beta_{kp} x_p \tag{2.6}$$

2.1.2 Rozhodovací strom

Rozhodovací strom je jedným z jednoduchších klasifikačných modelov. Pri pohľade na štruktúru, rozhodovací strom pozostáva zo série jednoduchých otázok a podľa nich rozhoduje o zaradení dát do tried.

Začiatok rozhodovacieho stromu je na vrchole. Rozdelenie dát je potrebné rozdeliť na takej vlastnosti (feature), ktorá poskytne najväčší informačný zisk. Tvorba rozhodovacieho stromu je iteratívny proces. Rozdelenie dát prebieha, pokým listy v uzloch nasledovníkov nebudú zaradené do rovnakej triedy. Problémom je však pri väčšom množstve vlastností hĺbka stromu a tým aj prílišné prispôsobenie algoritmu tréningovým dátam, tzv. „overfitting“. Rozhodovací strom je následne potrebné „osekať“ zadefinovaním maximálnej hĺbky stromu.
Maximalizácia informačného zisku pozostáva z definovania objektívnej funkcie. Objektívna funkcia je definovaná v rovnici č.2.7. Písmeno f v rovnici určuje vlastnosť, na ktorej delíme trénovacie dáta, D_p je dataset uzla rodiča a D_j je dataset uzla následovníka. I označuje koeficient nečistoty, N_p a N_j označuje počet trénovacích dát uzla rodiča a uzla nasledovníka. Informačný zisk je rozdielom medzi koeficientom nečistoty uzla rodiča a súčtom koeficientov nečistôt uzlov nasledovníkov. Čím menší

koeficient nečistoty uzlov nasledovníkov, tým je informačný zisk väčší. Väčšina machine learning knižníc má implementované binárne rozhodovacie stromy, teda uzol rodiča má iba dva uzly nasledovníkov.

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \tag{2.7}$$

Koeficient nečistoty gini (I_G), entropia (I_H) a klasifikačná chyba (I_E) sú bežne používané ako parametre rozdelenia dát.

Entropia je definovaná v rovnici č.2.8. $p(i|t)$ je pomer vzorov, ktoré patria triede i na počet vzorov uzlu t. Entropia je nulová, ak všetky vzory daného uzlu sú v jednej triede. Maximálnu hodnotu entropie (1) získame, ak pomer vzorov oboch tried pre binárny rozhodovací strom má rovnomerné rozloženie, teda napr. $p(i = 1|t) = 0.5$ a $p(i = 0|t) = 0.5$.

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t) \tag{2.8}$$

Gini koeficient nečistoty je kritérium na minimalizáciu pravdepodobnosti klasifikácie do zlej triedy. Gini koeficient je definovaný v rovnici č.2.9. Podobne ako entropia, koeficient gini má maximálnu hodnotu, ak majú triedy rovnomerné rozloženie.

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t) = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{2.9}$$

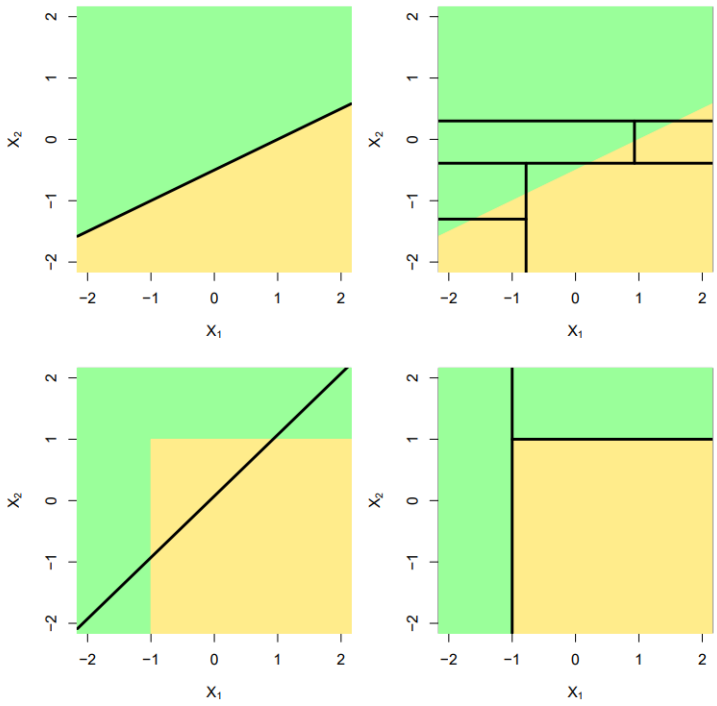
Klasifikačná chyba je kritérium vhodné na zmenšenie, resp. skrátenie rozhodovacieho stromu. V rovnici č.2.10 je definovaná klasifikačná chyba. Odčítaním pomeru vzorov triedy i ku všetkým vzorom v danom uzle t získame pomer klasifikovaných vzorov, ktoré sú nesprávne zaradené do triedy ku všetkým vzorom v danom uzle t. Nevýhodou klasifikačnej chyby je malá citlivosť na rast rozhodovacieho stromu. Znižovaním klasifikačnej chyby pomocou zväčšovania stromu dochádza k prílišnému prispôsobeniu algoritmu trénovacím dátam a tým k nesprávnej klasifikácii na budúcich dátach.[4]

Ak je cieľom vyhodnotiť čistotu nami určeného uzla, vhodnými kritériami sú gini koeficient a entropia, keďže sú citlivejšie na čistotu uzla ako klasifikačná chyba. Klasifikačnú chybu je vhodné využiť na zvýšenie presnosti predikcie rozhodovacieho stromu, ak už je upravená (zrezaná) výška.

$$I_E(t) = 1 - max(p(i|t)) \tag{2.10}$$

Dôležitým faktorom pri klasifikácii je výber správneho algoritmu podľa typu spracovávaných dát. Na obrázku č.2.3 je zobrazené porovnanie klasifikácie rôznych typov spracovávaných dát. Dáta zobrazené v hornej polovici obrázku majú rozhodovaciu

hranicu (decision boundary) lineárnu. Aplikácia jednoduchého algoritmu, napr. perceptron, je vhodnejšia ako rozdelenie dát (feature space) na niekoľko obdĺžnikov, ktoré používa rozhodovací strom. Situácia v dolnej polovici obrázku je iná. Dáta nie je možné lineárne separovať. Vhodnejším algoritmom je rozhodovací strom.[5]



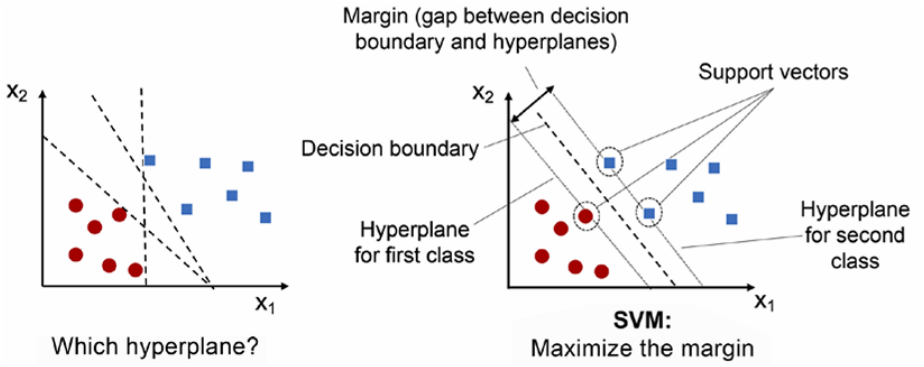
Obr. 2.3: Aplikácia algoritmov na lineárne a nelineárne separovateľné dáta[5]

2.1.3 Support vector machine algoritmus

Support vector machine (SVM) je model strojového učenia, ktorý dokáže klasifikovať lineárne aj nelineárne vzory v dátach. Úspešnosť klasifikácie modelu závisí od veľkosti datasetu. Algoritmus sa využíva najmä pri malých a stredne veľkých datasetoch (stovky až tisíce vzorov). [6]

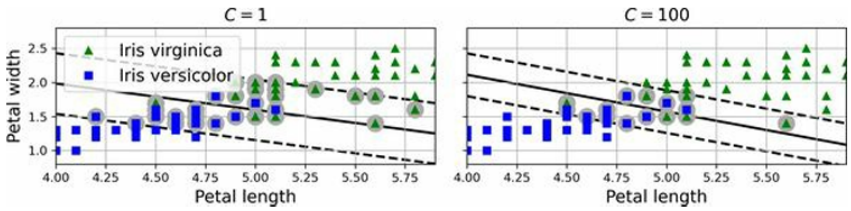
Pomocou SVM modelu oddeľujeme triedy rovinami, kde pomocou optimalizátora chceme dosiahnuť maximálnu možnú medzeru medzi triedami. Na obrázku č. 2.4 je zobrazené oddelenie tried pomocou SVM algoritmu. Body (vzory), ktoré ležia najbližšie k rozdeľovacej rovine, sa nazývajú podporné vektory (support vector), preto názov support vector machine.

Po určení veľkosti hranice môže nastať problém so vzormi z opačnej triedy, ktoré sú tesne pri podporných vektoroch. Možným riešením je zmenšenie medzery medzi triedami, čo ale vedie k zmenšeniu úspešnosti klasifikácie. Riešením je nájsť rovnováhu medzi maximalizovaním medzery medzi podpornými vektormi a braním



Obr. 2.4: SVM klasifikátor s rozhodovacími rovinami[4]

ohľadu na vzory, ktoré sa nachádzajú medzi nimi. Tzv. soft margin klasifikácia, ktorá pridáva parameter C do výpočtu medzery. Znižovaním parametru C zväčšujeme medzeru medzi podpornými vektormi a tým zmenšujeme šancu preučenia sa (overfitting) klasifikátora na poskytnuté dáta. Parameter C môžeme zaradiť medzi hyperparametre, ktorý kontroluje počet chýb klasifikátora. Porovnanie rozdielnych veľkostí C je zobrazené na obrázku č.2.5.[6][4]

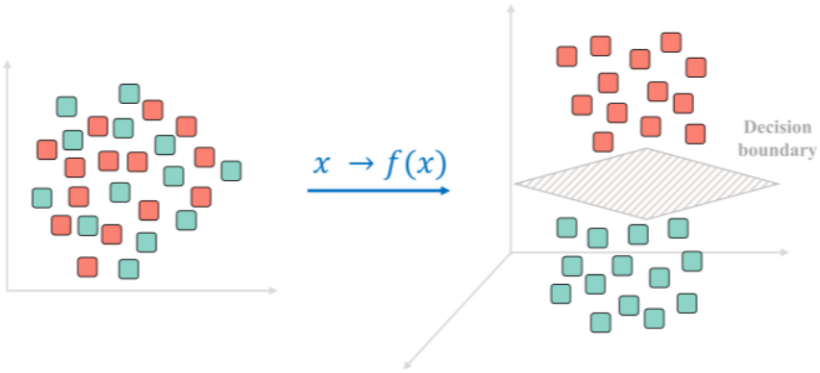


Obr. 2.5: Zmena parametra C v SVM klasifikátore[6]

SVM s jadrom dokážeme aplikovať na nelineárne dáta. Princípom metódy využívajúcej jadro je vytvorenie nelineárnych kombinácií príznakov (features) a ich následné premietnutie do vyššej dimenzie. V rovnici č.2.11 je zobrazená projekcia z 2D do 3D priestoru. Pomocou nelinearity $x_1^2 + x_2^2$, ktorá určuje vzdialenosť od stredu vytvoreného kruhu. Kruh definuje oddelovaciu rovinu.

$$\phi(x_1, x_2) = \left(x_1, x_2, x_1^2 + x_2^2\right) \tag{2.11}$$

Na obrázku č.2.6 je zobrazená projekcia z 2D do 3D roviny. Po projekcii stačí použiť lineárnu oddelovaciu rovinu, ktorá sa po spätnej projekcii do originálneho priestoru vlastností stane nelineárnou.



Obr. 2.6: Oddelenie pomocou projekcie do 3D roviny[17]

Postup projekcie do vyšších dimenzií je výpočtovo náročná úloha a preto sa používa tzv. Kernel trick. Pomocou tejto metódy nie je potrebné vytvárať nový viacdimenzionálny priestor nad priestorom vlastností a stačí vypočítať tzv. mieru podobnosti bodov. V rovnici č.2.12 je zobrazený výpočet podobnosti. Výpočtom euklidovskej vzdialenosti zistíme mieru podobnosti bodov. Parameter γ určuje, ako vzdialenosť bodov ovplyvní výslednú podobnosť. Tento parameter optimalizujeme pri učení modelu. [4]

$$\kappa(x^{(i)}, x^{(j)}) = \exp\left(-\gamma \left\| x^{(i)} - x^{(j)} \right\|^2\right) \tag{2.12}$$

2.1.4 Viac vrstvová neurónová sieť perceptron

Viacvrstvový perceptron (MLP) alebo inak nazývaný aj ako dopredná hlboká neurónová sieť je aproximátor určitej funkcie f. Sieť sa nazýva dopredná z dôvodu toku informácií zo vstupu, kde predávame vstupnú premennú x, následne sa vypočítajú medzivýsledky vedúce k definovaniu funkcie f k finálnemu výstupu y.Sieť s doprednýM učením mapuje funkciu $y = f(x; \theta)$ a počíta resp. učí sa parametre θ, ktorých úlohou je čo najlepšie aproximovať funkciu.

Model je možné definovať ako sériu po sebe idúcich transformácií. Pomocou rovnice č. 2.13 najskôr definujeme M lineárnych kombinácií vstupných parametrov $x_1, ..., x_D$ kde $j = 1, ...M$. Číslo jedna nad koeficientami znamená príslušnosť k prvej vrstve siete.Parameter $w_{ji}^{(1)}$ označuje váhu a parameter $w_{j0}^{(1)}$ označuje prah neurónu.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.13}$$

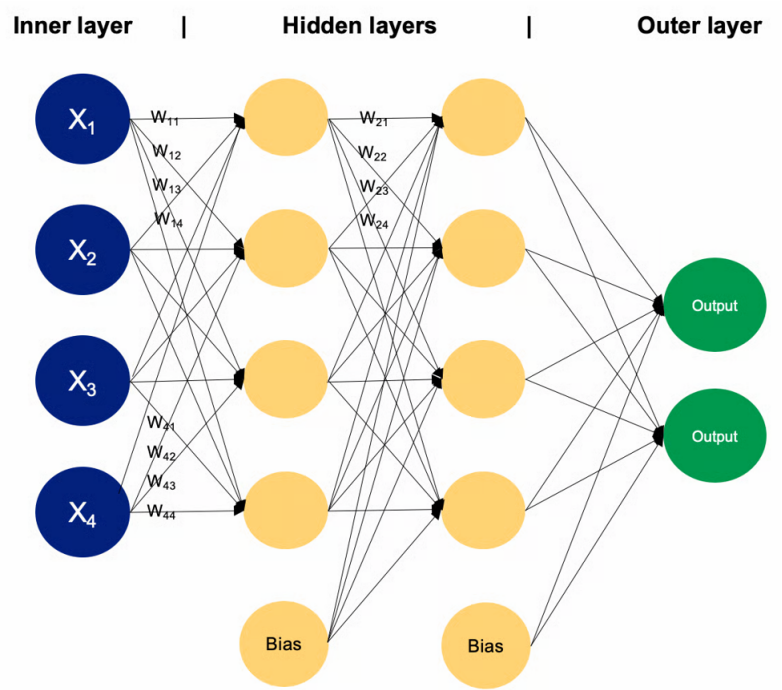
Parameter a_j sa nazýva aktivácia a vstupuje ako funkčná hodnota do aktivačnej funkcie $h(*)$ v rovnici č. 2.14.

$$z_j = h(a_j) \tag{2.14}$$

V nasledujúcej vrstve sú výstupné hodnoty z prvej vrstvy taktiež lineárne skombinované a vznikajú aktivácie pre výstupnú aktivačnú funkciu. Koeficient $k = 1, ..., K$ a K v rovnici č. 2.15 definuje počet výstupov. [13][14]

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.15}$$

Na obrázku č. 2.7 je zobrazená 3-vrstvová neurónová sieť a ako sú medzi sebou neuróny spojené. Zobrazená MLP sieť klasifikuje do viacerých tried.

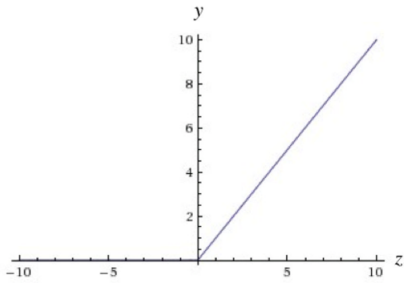


Obr. 2.7: Viacvrstvová neurónová sieť[15]

Aktivačné funkcie

Ako aktivačná funkcia na obrázku č.2.8 pre vnútornú vrstvu neurónovej siete je najčastejšie používaná ReLU funkcia. Jej predpis v rovnici č. 2.16 mapuje výstup z kladných hodnôt ako identitu a záporných hodnôt ako nulu.

$$f(z) = max(0, z) \tag{2.16}$$



Obr. 2.8: ReLU aktivačná funkcia[16]

Koeficient k v rovnici č.2.17 definujúcu softmax funkciu označuje počet výstupných neurónov. Softmax funkcia vychádza z Bayesovej teorémy, kde výsledok je daný súčinom podobnosti vzoru s trénovanými dátami a početnosťou podobných dát. Softmax funkcia prevedie aktiváciu na pravdepodobnosť zaradenia do triedy k.

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \frac{exp(a_k)}{\sum_j exp(a_j)} \tag{2.17}$$

Stratová funkcia opisuje rozdiel medzi predikciou modelu neurónovej siete a skutočnou hodnotou, ktorá má byť na výstupe. Pri klasifikátore s viacerými triedami používame stratovú funkciu s krížovou entropiou. V rovnici č. 2.18 je uvedený vzorec na výpočet stratovej funkcie pre viactriedny klasifikátor. Parameter E označuje vypočítanú stratovú funkciu, parameter K označuje počet tried, t_{nk} označuje one-hot encoding parameter, resp. či patrí daná vzorka do konkrétnej triedy, alebo nie. Parameter $y_k(\mathbf{x}_n, \mathbf{w})$ označuje vypočítanú pravdepodobnosť zaradenia do triedy.[14]

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(\mathbf{x}_n, \mathbf{w}) \tag{2.18}$$

Backpropagation technika je podstatná časť trénovania neurónových sietí. Jej úlohou je vypočítať gradient za pomoci spätného šírenia stratovej funkcie celou neurónovou sieťou. Princípom je výpočet derivácií stratovej funkcie vzhľadom na váhy siete. Ako prvé sa vypočíta derivácia chyby výstupnej vrstvy a následne sa vypočítané gradienty šíria až na vstup neurónovej siete. Tento krok sa nazýva reťazové pravidlo. Neurónová sieť obsahuje viacero neurónov, ktoré sú navzájom prepojené, to znamená, že ku každému neurónu vedie viacero ciest. Ak sa chceme dostať až na začiatok siete, tak je vhodné využiť jacobiho maticu, ktorá obsahuje všetky parciálne derivácie výstupov neurónov vzhľadom na vstupy.[13]

Regularizácia neurónovej siete sa využíva na zníženie hrozby preučenia neurónovej siete na konkrétne tréningové dáta. Pomocou parametru λ, ktorý v súčine s aktivačnou funkciou $f(\theta)$ pričíta k stratovej funkcii, obmedzíme veľkosť a rast váh.

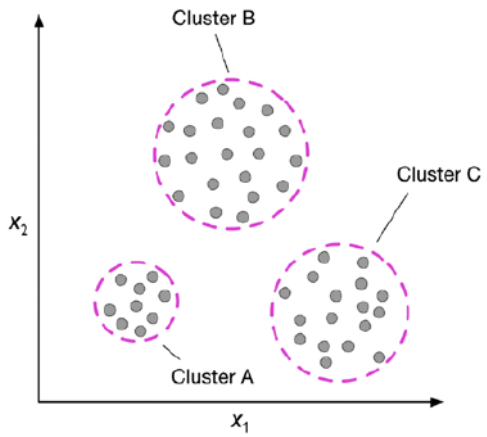
Ďalšou možnosťou je použitie dropout techniky. Neurón pri tejto technike pracuje s určitou pravdepodobnosťou p. Počas tréningu sa vypínajú neuróny s určitou pravdepodobnosťou. Touto technikou je neurónová sieť prinútená nespoliehať sa na zopár neurónov. [16]

2.2 Strojové učenie bez učiteľa

Pri strojovom učení s učiteľom poznáme výstupný signál predtým, ako natrénujeme model. Strojové učenie bez učiteľa pracuje s neoznačenými dátami alebo bez vopred známej štruktúry dát. Technikami strojového učenia bez učiteľa sme schopní skúmať štruktúru dát a následne získať cenné informácie bez známeho výsledku resp. výstupného signálu.

2.2.1 Zhlukovanie

Zhlukovanie je dátová analýza, ktorá skúma vzory v dátach. Táto technika nám dovoľuje zorganizovať resp. rozdeliť informácie v dátach do podskupín (zhlukov) bez vopred známych prepojení medzi dátami. V každom zhluku sa nachádzajú objekty, ktoré vykazujú určité podobnosti oproti ostatným objektom v iných zhlukoch. Zhlukovanie sa občas nazýva aj klasifikácia bez učiteľa. Na obrázku č.2.9 je vyobrazené jednoduché zhlukovanie do troch zhlukov. Využitie nachádza v niekoľkých odvetviach, napr. zákaznícka segmentácia, vyhľadávače, obrazová segmentácia, systémy na odporúčanie produktov, redukcia dimenzií v datasete.[4][6]



Obr. 2.9: Zhlukovanie[4]

K-means algoritmus

K-means algoritmus je jednoduché rozdelenie dát do K rozdielnych zhlukov, ktoré sa navzájom nekryjú. Na použitie algoritmu je potrebné zadefinovať potrebný počet zhlukov K. Následne algoritmus pridelí každý objekt v datasete do jedného z K zhlukov. Rovnica č.2.19 rieši prekrytie zhlukov nulovým prienikom.

$$C_k \cap C_{kn} = 0; k \neq kn \tag{2.19}$$

K-means algoritmus sa snaží dosiahnuť pri kvalitnom zhlukovaní, aby objekty vo všetkých zhlukoch mali čo najmenšiu rozdielnosť medzi sebou. V rovnici č.2.20 je definovaný parameter variability (rozdielnosti) $W(C_k)$. Výpočet variability pozostáva z podielu súčtu euklidovských vzdialeností medzi všetkými objektami v k-tom zhluku a celkovým počtom objektov $|C_k|$ v k-tom zhluku.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \tag{2.20}$$

Hľadaním najmenšieho súčtu všetkých parametrov variability $W(C_k)$ vzniká optimalizačný problém. Riešenie je ale výpočtovo náročné, keďže môže byť celkovo K^n iterácií.

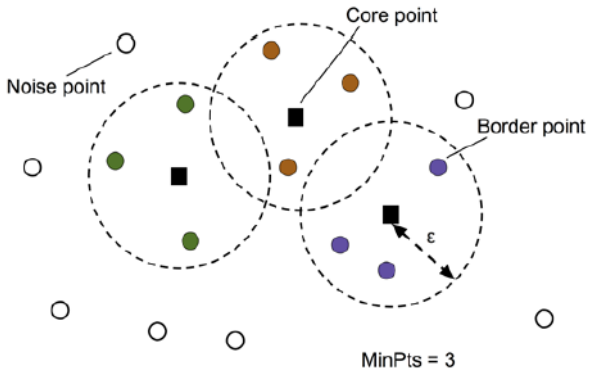
Riešením na vzniknutý problém je hľadanie lokálneho optima pridelením každému objektu číslo od 1 po K. Druhým krokom je výpočet centroidu pre každý zhluk a následne sa každý objekt pridelí najbližšiemu centroidu. Najbližší centroid je definovaný euklidovskou vzdialenosťou. Druhý krok iterujeme, pokým sa vzniknuté stredy zhlukov neprestanú meniť.

Podstatným krokom pri správnom zhlukovaní je viacnásobné spustenie algoritmu. Pre K-means platí, že hľadá lokálne optimum, nie globálne optimum. Až niekoľko spustení zaručí konvergovanie ku správnemu zhlukovaniu.[5]

DBSCAN

Density based spatial clustering of aplications with noise (DBSCAN) je zhlukovací algoritmus vyhľadávajúci miesta v dátach, kde je vyššia hustota v rámci špecifikovaného okruhu ϵ. Dôležitou premennou je minimálny počet susedov daného objektu.

Objektom v datasete sú pridelené označenia. Ak má objekt v okolí ϵ počet susedov rovný alebo väčší ako definovaný minimálny počet susedov, označujeme ho ako jadro. Za krajný objekt sa považuje objekt s počtom susedov menším, ako je definovaný minimálny počet susedov. Ostatné body sú považované za šum. Na obrázku č.2.10 je zobrazený vznik jadra a krajného objektu. Krajný objekt je pridelený do zhluku, ktorého jadro spadá pod okruh ϵ.



Obr. 2.10: DBSCAN algoritmus[4]

Jednou z výhod DBSCAN algoritmu je odlišný prístup ku tvaru zhlukov ako pri K-means algoritme, kde je použité kruhové okolie zhluku. Nevýhodou je viac parametrov ako K-means a tým aj zložitejšie hľadanie optimálnych hodnôt pri väčšom datasete.[4]

2.2.2 Odhad hustoty

Odhad hustoty je súbor metód na určenie hustoty pravdepodobnosti procesu, ktorý vytvoril dataset resp. spracovávané dáta. Metóda sa využíva najmä na vyhľadávanie anomálií.[6]

Odhad hustoty jadra Odhad hustoty jadra je neparametrická metóda, ktorá použitím datasetu odhaduje pravdepodobnosť nových bodov. Hustota je priamo úmerná počtu bodov v okolí práve počítaného bodu. Jadrová funkcia mapuje vektory (dáta z datasetu) x a x_i do priestoru bodov K(hodnota jadra). Jadro definuje podobnosť medzi vektorom x a vektorom x_i. Čím väčšia podobnosť, tým je hodnota K vyššia. V jadre sa nachádza parameter šírka pásma, ktorá kontroluje variáciu(hladkosť) priebehu. [11]

Existuje niekoľko typov jadier. Medzi najpoužívanejšie patria Gaussove jadro alebo obdĺžnikové jadro.

V rovniciach č.2.21 a 2.22 je definované Gaussove a obdĺžnikové jadro. Parameter u je definovaný ako rozdiel x a x_i.

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-0.5u^2} \tag{2.21}$$

$$K(u) = \frac{1}{2} \tag{2.22}$$

V rovnici č.2.23 je definovaná funkcia odhadu hustoty jadra. n parameter je počet prvkov v datasete, parameter h je šírka pásma.[12]

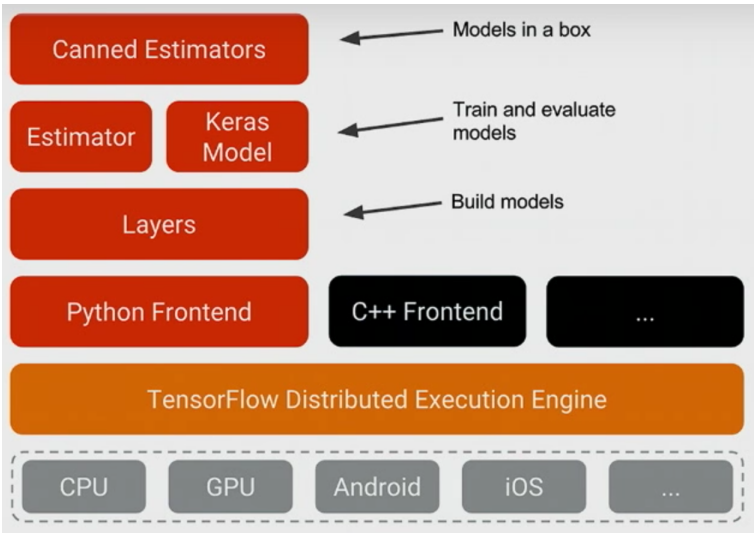
$$f_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{2.23}$$

3 Praktická časť

V tejto časti som vypísal mnou vytvorené programy. Postupoval som od tých jednoduchších programov na rozdelenie datasetu ku klasifikácii. Ako programovací jazyk som zvolil Python, pre ktorý je vytvorené veľké množstvo knižníc a sú voľne dostupné. Najviac som využil knižnicu scikit-learn a tensorflow knižnicu, ktorá obsahuje modely keras. Jeden program (klasická metóda oddelenia) je realizovaný v prostredí matlab.

Použité softvérové knižnice

Scikit learn je open source knižnica pozostávajúca z algoritmov strojového učenia. Konkrétne obsahuje algoritmy na klasifikáciu, regresiu, zhlukovanie alebo nástroje na úpravu dát. Knižnica je určená na použitie v programovacom jazyku Python.
Tensorflow Tensorflow je knižnica určená primárne na tvorbu neurónových sietí, ale aj na aplikácie strojového učenia. Je to jeden z najpoužívanejších frameworkov v súčasnosti. Použitie je možné v niekoľkých programovacích jazykoch, ako napr. C++ alebo Python. Knižnica obsahuje aj Keras modely, ktoré sú tzv. frontend pre knižnicu Tensorflow, ktorá zabezpečuje optimalizáciu, vytvorenie topológie modelu alebo výpočty váh. Keras API je súborom niekoľkých typov neurónových sietí alebo nástrojov. Príkladom neurónových sietí môže byť Perceptron alebo Transformátor.



Obr. 3.1: API knižnice Tensorflow

3.1 Vstupné Dáta

Vstupné dáta sú vo forme tabuľky s niekoľkými stĺpcami, s ktorými som pracoval priamo, a ostatné som použil ako referenciu k overeniu funkčnosti oddelenia a klasifikácie vozidiel.

Dáta, ktoré som použil na tréning resp. klasifikáciu sú získané priamo z nápravových senzorov umiestnených na vozovke. Súbor sa nazýva *AxleEventRecords* a je vo forme tabuľky so 14 stĺpcami, ktoré obsahujú (uvedené sú len použité stĺpce):

- channel-kanál senzoru, číslovanie je zvyčajne od 0 a do počtu senzorov na vozovke. Môže byť ale aj iné vnútorné označenie senzorov napr. 11 a 13.
- frame_start- ID konkrétneho frameu, kedy bol zaznamenaný vstup kolesa na senzor
- sample_start- ID konkrétneho vzorku, kedy bol zaznamenaný vstup kolesa na senzor
- time_start a microsec_start- konkrétny čas, kedy bol zaznamenaný vstup kolesa na senzor
- frame_center- ID konkrétneho frameu počas prechodu ťažiska kolesa cez senzor
- sample_center- ID konkrétneho vzorku počas prechodu ťažiska kolesa cez senzor

Výpočty som uskutočňoval použitím hodnôt zo stĺpcov *frame_start* a *sample_start*. Dôvodom je nestabilita času a tým zhoršená presnosť merania.

Frame pozostáva z 512 vzoriek a veľkosť 1 vzorky je závislá na frekvencii vzorkovania. V mojom prípade je veľkosť vzorkovania 10 kHz, ale tento parameter je len pre dodaný dataset a frekvencia vzorkovania bude pri použití v premávke nižšia.

source	channel	frame_start	sample_start	time_start	microsec_start	frame_center	sample_center	time_center	microsec_center
0	0	26	251	########	407500	26	298	#########	412200
0	1	27	397	########	473300	27	445	#########	478100
0	2	29	32	########	539200	29	77	#########	543700

Obr. 3.2: Ukážka vstupných dát

Kontrolné dáta som získal z metodiky WIM(weight in motion). Dáta boli získané pomocou indukčných sľučiek a nemusia byť úplné, alebo mohlo dôjsť k spojeniu vozidiel, resp. oddeleniu vozidla od prívesu a následne k nesprávnej klasifikácii. Riešenia na spomenuté chyby budú aplikované v mojich klasifikačných algoritmoch.

Dôležité stĺpce sú nasledovné:

- datetime,ms-čas vstupu vozidla na senzor(indukčná sľučka)
- class-zaradenie do konkrétnej triedy vozidla podľa klasifikačnej schémy(WIM schéma 64 tried)

- velocity- rýchlosť vozidla
- length- dĺžka vozidla v decimetroch
- axle_count- počet náprav vozidla
- axle_X_Y- rázvory medzi nápravami v centimetroch

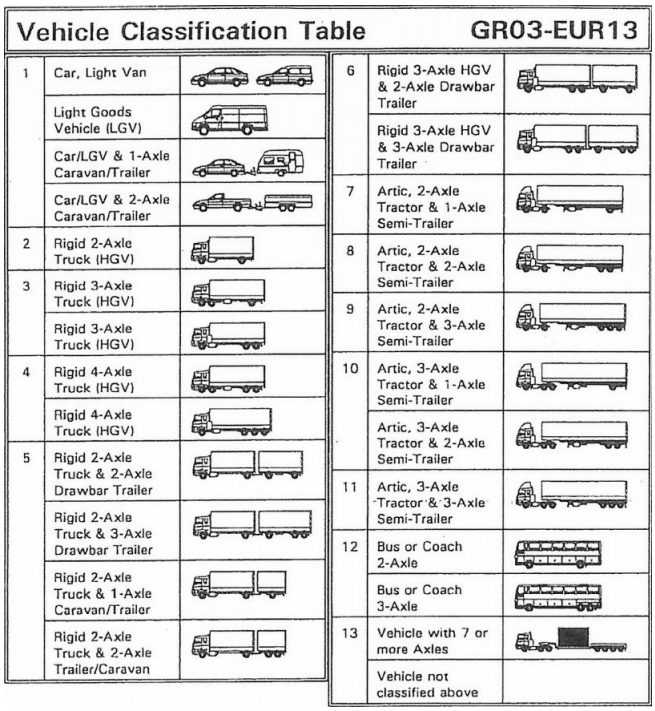
datetime	ms	wim	class	loop_class	velocity	length	total_weight	axle_count	axle_1_2_	axle_2_3_	axle_3_4_	axle_4_5_	axle_5_6_	axle_6_7_	axle_7_8_
########	253	KP-CE-W4	23	8	74	169	20600	5	382	574	131	132	0	0	0
########	548	KP-CE-W4	23	8	77	168	41300	5	361	558	132	132	0	0	0

Obr. 3.3: Ukážka kontrolných dát

3.2 Klasifikačné schémy

3.2.1 EUR13

Každý štát alebo región má svoju klasifikačnú schému. V Európe je zaužívaná tzv. tabuľka EUR13 vďaka implementácii väčšiny vozidiel, ktoré jazdia na kontinente. Kategoricky je rozdelená na 13 rôznych typov vozidiel s dôrazom na nákladné vozidlá.



Obr. 3.4: EUR13 tabuľka

3.2.2 WIM

Klasifikácia 64 triedami je používaná pri vážení pri pohybe(WIM). Rozšírenie na viacero tried je presnejšou variantou. Dokáže klasifikovať rôznorodé vozidlá a tým zabezpečí vyššiu presnosť pri použití klasifikátora napr. na výber mýta v danom úseku.

V mojich vlastných programoch som použil prevod tabuľky WIM(64 tried) na EUR13. Ukážka prevodu je na obrázku č.3.5. Prevod je určený podľa podobnosti vozidiel vhodných do danej triedy.

WIM	Description	Picture	EUR 13 RSD/GR03
-100	Category for special use	!	13
0	Vehicle not classified		13
1	Car		1
2	Van		1
3	Car 1-Axle Trailer		1
4	Car 2-Axle Trailer		1
5	2-Axle Truck		2
6	3-Axle Truck		3
7	3-Axle Truck		3

Obr. 3.5: Prevod EUR13 na WIM

3.3 Pozdĺžna separácia vozidiel

3.3.1 Konvenčné spracovanie náprav

Konvenčné spracovanie náprav je využitie základných fyzikálnych poznatkov, ako je výpočet dráhy, rýchlosti a zrýchlenia, a následné porovnávanie týchto dát medzi sebou. Program som realizoval v prostredí Matlab.

Ako prvé som vypočítal časový úsek od prvej vzorky. Jednotlivé dáta idú za sebou a jednoduchým výpočtom uvedeným v nasledujúcom výpise, kde *t0* znázorňuje počiatočný bod, ktorý je odčítaný od každého ďalšieho bodu v datasete. Jednotlivé časové vzorky sú v slede za sebou a preto je potrebný výpočet od absolútneho začiatku. Siedmy stĺpec v poli *data* obsahuje konkrétny frame a ôsmy stĺpec presnú vzorku, kedy koleso bolo v strede senzora.

```
ch(channel,ch_id(channel)) =data(i,7)*512+data(i,8) - t0;
```

Ďalším krokom je výpočet rýchlosti z vypočítaných časových udalostí v minulom kroku. V nasledujúcom výpise vypočítavam rýchlosť jednoduchým fyzikálnym

vzorcom. Výpočet rýchlosti je vytvorený priamo pre prvý dataset, ktorý obsahuje 3 senzory.

```
    v(1,i) = dist/(ch(2,i)-ch(1,i))*36000; %km/h
    v(2,i) = dist/(ch(3,i)-ch(2,i))*36000; %km/h
```

Ďalším krokom je iterácia cez uložené pole s časmi vstupov a následný výpočet vzdialeností medzi nápravami pomocou časov medzi nápravami,rýchlosťami a vzdialenosťou medzi senzormi. Aby sa vozidlá dali oddeliť, je potrebné stanoviť určité pravidlá. Podľa požiadaviek som stanovil niektoré parametre ako napr. rýchlosť. Minimálne,maximálne vzdialenosti náprav alebo aj maximálny počet náprav som zistil pomocou tabuľky EUR13. Zrýchlenie a spomalenie som určil odhadom približne na 1g. Následným porovnávaním s maximálnymi/minimálnymi hodnotami som vytvoril hranice medzi vozidlami.

```
min_v = 5;     %km/hod  %minimalna rychlost
max_v = 250;   %km/hod  %maximalna rychlost
min_a_d = 50; %cm%minimalna vzdialenost medzi napravami
max_a_d = 1501;%cm%maximalna vzdialenost medzi napravami
max_a_num = 18;% maximalny pocet naprav
max_acc = 10;  %max zrychlenie
min_acc = -10; %max spomalenie
```

Posledným krokom je použitie vytvorených hraníc medzi vozidlami ako podmienky a jednoduchou iteráciou cez pole s uloženými rázvormi som oddelil vozidlá.

Zhodnotenie metódy

Pri pohľade na vytvorený dataset (priložený v elektronickej prílohe) som nenašiel nespracované vozidlá alebo nezmyselne generované dáta. Problém nastáva pri ďalších datasetoch, ktoré obsahujú vozidlá bližšie k sebe. Výpočet rýchlosti generuje nezmyselné hodnoty, príčinou je viac šumu v datasetoch, resp. zachytené udalosti prechodu vozidla obsahujú iba polovicu alebo časť celkového počtu prechodov.

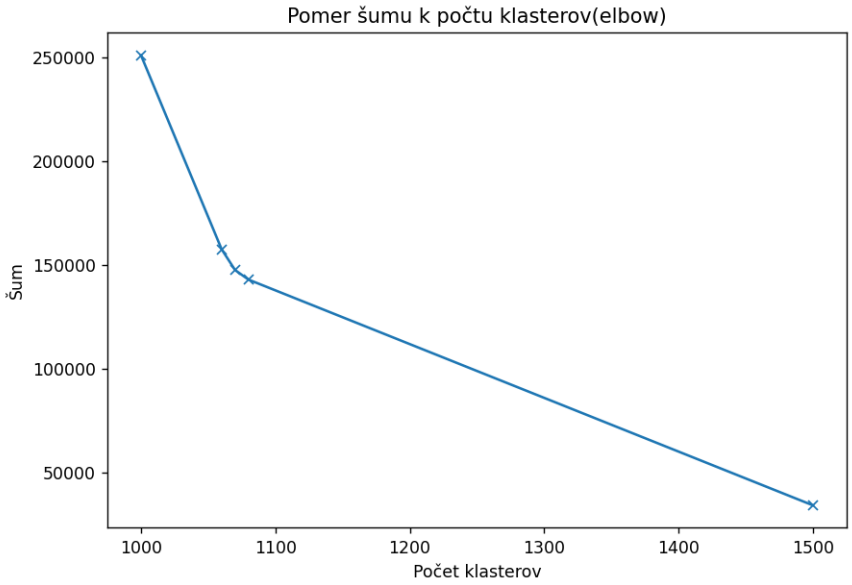
3.3.2 K-means algoritmus

Problémom pri zhlukovaní pomocou K-means algoritmu je nutnosť stanoviť počet zhlukov. Keďže ale nepoznám, aké vozidlá triedim resp. počet vozidiel je neznámy, musel som využiť heuristickú techniku na stanovenie počtu zhlukov. Nazýva sa metóda ohybu(elbow method) a jej funkciou je nájsť správny pomer nesprávne klasifikovaných dát k počtu zhlukov.

Na výpočet počtu zhlukov som použil metódu *.inertia_*. Jej funkciou je vypočítať súčet vzdialeností umocnených na druhú všetkých bodov k ich centrám zhluku.

Požiadavkou je, aby číslo bolo čo najmenšie. Pri nulovej inercii(zotrvačnosti) je ale každý bod v datasete samostatným zhlukom a to je z pohľadu potreby oddelenia dát kontraproduktívne [10].

Pomocou cyklu *for* som testoval rôzne počty zhlukov. Prvým pokusom bolo číslo 9000, ktoré som stanovil podľa počtu vstupných signálov(výstupy zo senzorov) v datasete. Následne som vyhotovil zoznam, ktorý som zaplnil hodnotami od 1000 do 9000, testoval ich na dátach a hľadal ohyb v grafe. Na obrázku č.3.6 je zobrazený pomer šumu k počtu zhlukov. Postupným zmenšovaním intervalu možných počtov zhlukov a hľadaním tzv. lakťov v priebehoch som získal číslo 1070.



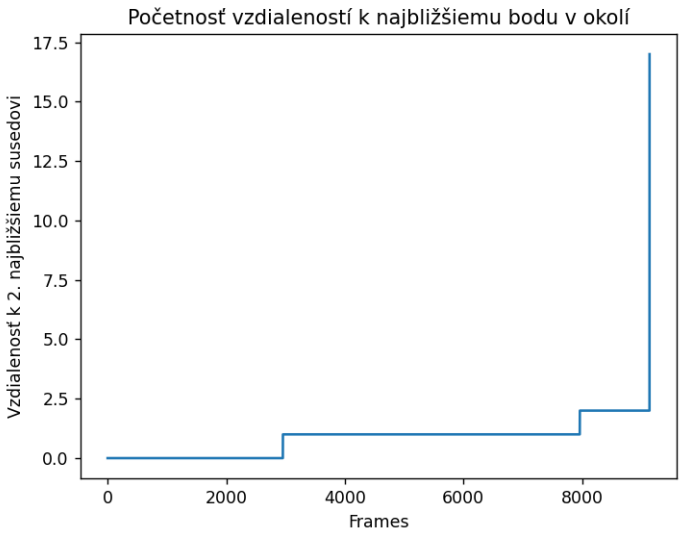
Obr. 3.6: Metóda ohybu

Jednou z nevýhod K-means algoritmu je počiatočné umiestňovanie centroidov. Pri hľadaní počtu zhlukov som tak musel niekoľkokrát spustiť program, kvôli nesprávnej počiatočnej pozícii centroidov. Tento jav je spôsobený náhodným výberom miesta centroidov na začiatku a pre správnu funkciu v prevádzke to je závažná chyba. Ďalšou nevýhodou je konverzia dát z 1D na 2D pre správne fungovanie algoritmu. Časová náročnosť algoritmu je oproti ostatným realizovaným zhlukovacím algoritmom väčšia kvôli testovaniu počtu zhlukov.

3.3.3 DBSCAN

Algoritmus DBSCAN som realizoval vlastným návrhom funkcie a aj použitím knižnice Sci-kit.

Prvým krokom pri zhlukovaní bolo stanovenie parametru *eps*. Prvým pokusom bolo zistiť parameter pomocou nájdenia zlomu v histograme. Využil som triedu *NearestNeighbours* na zistenie vzdialenosti bodu k najbližšiemu susednému bodu. Vzdialenosti som zoradil od najmenšej po najväčšiu a zobrazil som ich na obrázku č.3.7.



Obr. 3.7: Početnosť vzdialeností k najbližšiemu bodu v okolí

Bod zlomu je podľa priebehu na hodnote 2. Po použití tejto hodnoty ako parameter epsilon je ale zhlukovanie neúčinné. Vozidlá, najmä kamióny a ďalšie nákladné vozidlá, ktoré majú nápravy vzdialené aj 15 metrov, sú rozdelené do viacerých zhlukov. Nízkou hodnotou epsilon je ovplyvnené celé zhlukovanie a aj klasifikácia do tried. Parameter som začal zvyšovať po jednotkách. Body, ktoré sú brané ako šum, sú označené hodnotou -1. Zvolil som hodnotu 8.

3.3.4 Vlastný DBSCAN algoritmus

Prvým krokom bola úprava dát na 1D pole. Algoritmus iteruje cez celý dataset a porovnáva aktuálny bod s predošlým bodom zväčšeným o parameter epsilon. Ak je vzdialenosť aktuálneho bodu väčšia ako parameter epsilon, tak aktuálny zhluk sa uloží a vytvorí sa nový zhluk.

Parameter epsilon som využil najskôr z predošlého kódu, kde sa realizovalo zhlukovanie pomocou knižnice DBSCAN. Zvolil som druhú alternatívu, kde som menil parameter epsilon podľa dodaného datasetu. Kód sa líši od predošlého výpisu aj tým, že spracovávam ďalšie dáta. Tento algoritmus som použil pri klasifikácii, ktorá je riešená v nasledujúcej sekcii.

3.3.5 Spojenie K-means a DBSCAN

Problémom pri K-means algoritme je, že nepoznám koľko zhlukov obsahujú neznáme dáta.Vypočítaním zhlukov pomocou algoritmu DBSCAN som získal celkový počet zhlukov, ktoré obsahujú vozidlá. Následným dosadením počtu zhlukov som vypočítal zhluky pomocou K-means algoritmu. Týmto krokom som skombinoval výhodu DBSCAN (nepotrebujem poznať počet zhlukov) a K-means algoritmu (časové vzdialenosti medzi vozidlami). Jednou z nevýhod je potreba poznať parameter ϵ pri algoritme DBSCAN.

3.3.6 Odhad hustoty jadra kernel density estimation algoritmom

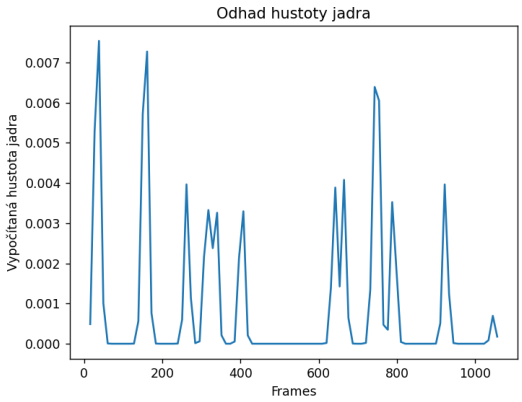
Techniku odhadu hustoty som použil najmä kvôli inému pohľadu na spracovávané dáta oproti zhlukovaniu. Odhad hustoty jadra som vyskúšal realizovať vlastnou implementáciou algoritmu.

Prvým krokom bolo vytvorenie vektora, ktorý má dĺžku maximálnej hodnoty bodu z dodaného datasetu rozšírený o 20 a počet bodov je získaný z celkového počtu bodov datasetu.

Pomocou dvoch cyklov for som vytvoril iterátor cez všetky body datasetu. Porovnával som aktuálnu hodnotu x s ostatnými hodnotami v datasete cez premennú xi. Vypočítal som gaussovské jadro a všetky hodnoty sčítal. Jadro gauss som zvolil kvôli tomu, že rozloženie dát nepoznám. Posledným krokom je hľadanie lokálneho minima. Šírku pásma bw som najskôr vybral ako náhodnú hodnotu. Postupne som ju znižoval, aby bola dosiahnutá vysoká variancia priebehu kvôli dátam, ktoré sú tesne pri sebe.

Odhad hustoty jadra vytvorený iteráciou cez celý dataset je veľmi výpočtovo náročný proces. Vhodnejším spôsobom je vytvorenie rekurzívneho stromu a tým znížiť celkovú časovú zložitosť. Na obrázku 3.8 je zobrazený priebeh odhadu hustoty jadra. Priebeh má vysokú varianciu z dôvodu nižšej hodnoty parametru bw. Na obrázku 3.9 sú vypísané lokálne minimá priebehu. Výsledné lokálne minimá som porovnal s kontrolnými dátami. Každé lokálne minimum je stred časového úseku, kedy cez merané miesto nešlo žiadne vozidlo.

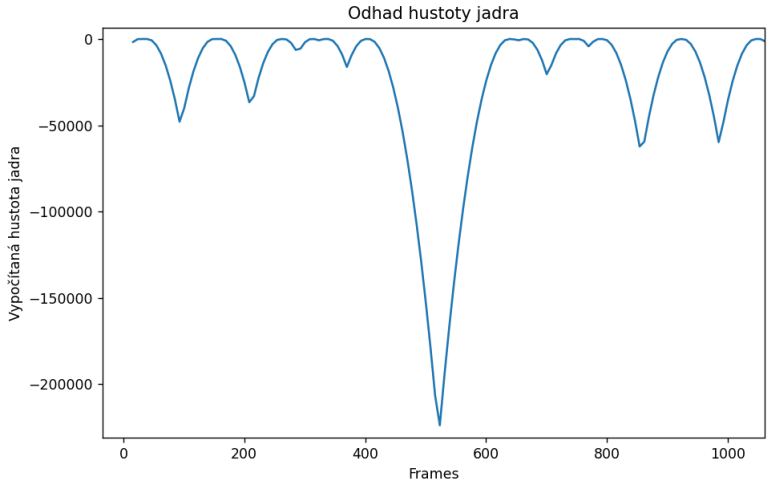
Na obrázku č.3.10 a 3.11 je zobrazený odhad hustoty jadra na prvých 1000 frameoch a k tomu prislúchajúce lokálne minimá.



Obr. 3.8: Odhad hustoty jadra vlastným algoritmom na prvých 1000 frameov

```
[ 94.27956989 206.10752688 284.38709677 329.11827957 373.84946237
 519.22580645 653.41935484 698.15053763 776.43010753 854.70967742
 988.90322581]
```

Obr. 3.9: Lokálne minimá



Obr. 3.10: Odhad hustoty jadra na prvých 1000 frameov

```
 92.91179689   208.27949223   285.19128912   323.64718757
369.7942657    523.61785949   654.3679142    700.51499234
746.66207047   769.73560954   854.33858612   985.08864084
```

Obr. 3.11: Lokálne minimá

3.3.7 Vyhodnotenie separácie vozidiel

Pomocou viacerých algoritmov na separáciu vozidiel som dosiahol rozličné výsledky. Celkový počet vozidiel z WIM je 3669. Konvenčná metóda bola využitá iba pre prvý dataset obsahujúci menej šumu a iných anomálií, preto je porovnanie s ostatnými algoritmami skôr irelevantné. Pri K-means algoritme, ktorý je spojený s metódou DBSCAN je počet separovaných vozidiel rovnaký ako pri DBSCAN metódach. Pre spomenutú metódu bez algoritmu DBSCAN som nenašiel vhodný spôsob, ako kvantifikovať počet rozdelených vozidiel. Oddelenie vozidiel pomocou odhadu hustoty jadra sa mi nepodarilo optimalizovať natoľko, aby sa tento algoritmus dal použiť v prevádzke. Preto som ho vyradil z porovnania.

Model	**Počet nájdených vozidiel**	**podiel nájdených vozidiel**
Konvenčná metóda	1057	98,7 %
DBSCAN	3662	99,8
Vlastný DBSCAN	3662	99,8 %
DBSCAN+K-means	3662	99,8 %

Tab. 3.1: Porovnanie algoritmov separácie vozidiel

Algoritmy, ktoré porovnávam, majú prakticky rovnakú úspešnosť oddelenia vozidiel. Rozdielna je však komplexnosť výpočtu. K-means aj DBSCAN sú z knižnice scikit-learn a sú primárne určené pre viacrozmerné dáta. Spojenie dvoch spomínaných algoritmov je kombináciou výhod oboch algoritmov, zmenila sa však aj výpočtová náročnosť. Mnou realizovaný DBSCAN je navrhnutý na jednoduché oddelenie prvkov v poli, ktoré so sebou nesusedia, a preto som si ho vybral ako algoritmus, ktorý použijem pri príprave datasetov a pri klasifikácii.

3.4 Spracovanie datasetu

Po oddelení vozidiel jedným z algoritmov zhlukovania sú vstupné dáta vo forme:

- senzory -na každom riadku je uvedená sekvencia impulzov na senzoroch pre každé vozidlo
- frame, sample - každý riadok patrí jednému vozidlu

Súbor sa rozdelí na 3 časti uvedené vyššie a následne sa prejde na výpočet rýchlosti a rázvorov. Prvým krokom je výpočet doby trvania medzi vstupom prvej nápravy a nasledujúcou nápravou. Sčítaním časových vstupov na prvý senzor po sebe idúcich náprav som zistil časovú vzdialenosť. Výpočet rýchlosti som realizoval pomocou vstupu prvej nápravy na prvý senzor a následne taktiež prvej nápravy na nasledujúci senzor. Ďalej som vypočítal rýchlosť pomocou jednoduchého vzorca na

výpočet rýchlosti (dráha za určitý čas). Rýchlosti som vypočítal 2 a to z dôvodu spriemerovania rýchlostí. Posledným krokom bolo znovu použitie vzorca na výpočet rýchlosti s tým, že som vypočítal neznámy rázvor pomocou vypočítaného času medzi za sebou idúcimi nápravami a vypočítanou rýchlosťou.

Prvým pokusom bolo porovnanie rázvorov vozidla s tabuľkou EUR13. Komparátor *cmp* porovnáva počet náprav s tabuľkou a vyhodnotí, do ktorých tried spadá vozidlo. Tento spôsob je ale nevhodný kvôli neznámemu pôvodu tabuľky. Vhodnejšou metódou je vyhľadanie konkrétneho vozidla vo WIM kontrolných dátach a vytvorenie labelu. V nasledujúcom výpise vyhľadávam výsledné triedy vozidiel v kontrolných dátach pomocou porovnávania vypočítaných rázvorov a následnou kombináciou s jednoduchými podmienkami. Jednotlivé triedy vozidiel sú rozdelené do 64 WIM kategórií. Pre konvertovanie som vytvoril lookup tabuľku s prepočtom na kategórie EUR13.

```
mask = (
       (dfinal['axle_count'] == df2.iloc[i,0]) &
              (dfinal['axle_1_2_base'].
              between(df2.iloc[i,1] - tolerance,
              df2.iloc[i,1] + tolerance))&
              (dfinal['axle_2_3_base'].
              between(df2.iloc[i,2] - tolerance,
              df2.iloc[i,2] + tolerance))&
              (dfinal['axle_3_4_base'].
              between(df2.iloc[i,3] - tolerance,
              df2.iloc[i,3] + tolerance))&
              (dfinal['axle_4_5_base'].
              between(df2.iloc[i,4] - tolerance,
              df2.iloc[i,4] + tolerance)))
```

Pred samotným tréningovým procesom som musel upraviť vstupné dáta do formy vhodnej na vstup do modelu klasifikátora. Pomocou vytvorenej funkcie *process_axle_data* v nasledujúcom výpise som konvertoval vstupný dataset do jednoduchého zoznamu. Následne som našiel a uložil do premennej *max_len* počet stĺpcov v datasete a vytvoril tzv. padding. To znamená zistenie najväčšieho počtu stĺpcov v datasete a následné doplnenie ostatných, kratších, riadkov nulami. Posledným krokom bolo uloženie datasetu do dátového typu touple.

Týmto krokom som zabezpečil rovnakú šírku vstupných dát do modelu, keďže rozmery vstupných dát sú pevne dané pri tréningu. Kvôli rôznym šírkam resp. počtom náprav je ťažšie docieliť uniformnosť vstupov, a preto je to jeden z najdôležitejších krokov k príprave datasetu.

```
def process_axle_data(ds_x):
    axle_data = []
    for index, row in ds_x.iterrows():
      axle_data.append(row.values.tolist())
    max_len= max(len(inner) for inner in axle_data)+2
    padded_list = [inner + [0] *
        (max_len-len(inner)) for inner in axle_data]
    padded_array = np.array(padded_list)
    # Extrahujeme všetky stlpce do jednej n-tice
    columns = tuple(padded_array[:, i]
            for i in range(max_len))
    return  X
```

Augmentácia Dát SMOTE technikou

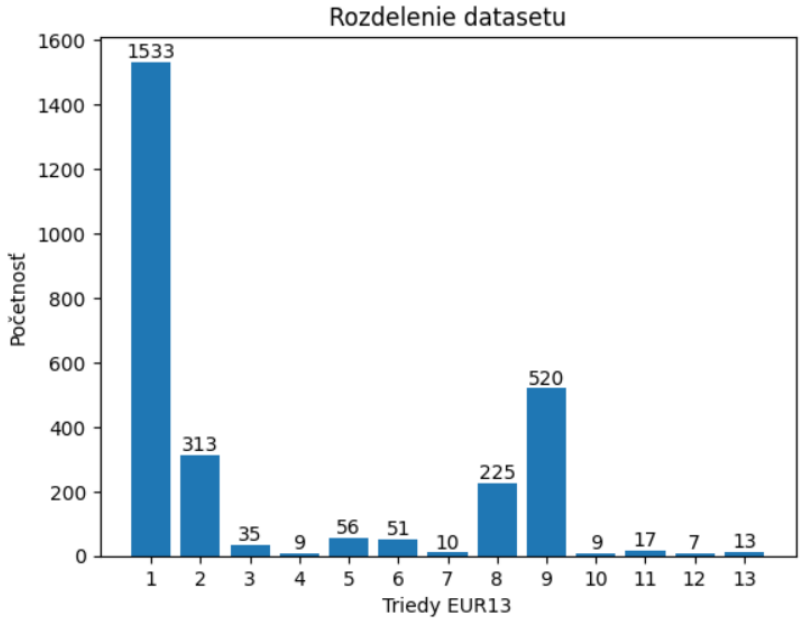
Učenie každého klasifikátora je ovplyvnené tým, aké dáta mu dodáme. Celosvetová doprava je zložená primárne z osobných vozidiel a dosiahnuť vytvorenie vyrovnaného datasetu je bez zásahu prakticky nemožné. Ako vhodnú metódu na umelé rozšírenie dát som si vybral tzv. SMOTE (Synthetic Minority Oversampling Technique), v preklade "Technika umelého navýšenia počtu vzoriek minoritných tried".Ak by sa klasifikátor naučený na nevyrovnanom datasete použil v lokalite s väčším výskytom nákladných, poprípade iných špeciálnych vozidiel, tak by došlo k nespoľahlivému zaradeniu do tried.

Pri pohľade na rozloženie počtu vzoriek na obrázku č.3.12 je vidieť veľkú prevahu osobných vozidiel a s odstupom aj triedy 2, 8 a 9.

Základným princípom SMOTE techniky je vybrať náhodne vzorku z jednej triedy a v okolí bodu nájsť susedný bod. Následne lineárnou kombináciou (interpolácia) získame bod v priestore vlastností, ktorý môžeme použiť ako vzorku do datasetu. Metóda je obsiahnutá v knižnici Imbalanced-learn vychádzajúca z knižnice scikit-learn.

Odporúčaným postupom je podvzorkovať najčastejšiu triedu v datasete a následne umelo zvýšiť ostatné triedy na jej úroveň. Ja som ale podvzorkovanie nespravil kvôli obavám z nesprávnej klasifikácie tried 1 a 2, ktoré sú si veľmi podobné, keďže v triedach figurujú osobné autá a dodávky.

V nasledujúcom výpise dopĺňam vzorky do datasetu pomocou SMOTE. Najskôr som musel upraviť podmienku počtu susedov v danej triede. Je to dané tým, že som nepoznal zloženie datasetu, a preto som využil dynamicky meniaci sa parameter *k_neighbors*.



Obr. 3.12: Početnosť v konkrétnych triedach datasetu

Na posledných dvoch riadkoch je uvedená metóda, kde som nastavil parameter *sampling_strategy*. Parameter je možné nastaviť troma spôsobmi:

- auto -automaticky zvolí počet vzoriek v triedach podľa najpočetnejšej triedy
- float - pomerom stanoví počet vzoriek(len pre binárny klasifikátor)
- dict - slovníkom určím ktoré triedy budú mať koľko vzoriek
- str - môže doplniť iba najmenšiu triedu alebo doplní všetky

Ako najvhodnejší som zvolil *auto* spôsob. Vhodnou alternatívou je aj doplnenie *dict* s módom doplnenia podľa požiadaviek, ktorá trieda je ako efektívna pri následnej klasifikácii.

```
min_samples = min(len(final_axles[final_axles == label])
            for label in np.unique(final_axles))
k_neighbors = min(5, min_samples - 1)
k_neighbors = max(1, k_neighbors)
smote = SMOTE(random_state=42,sampling_strategy='auto',
                k_neighbors=k_neighbors)
X, final_axles = smote.fit_resample(X, final_axles)
```

Vstup do klasifikátorov

Výsledným vstupom do klasifikátorov sú rázvory vozidiel s maximálne 8 nápravami. Ako label sú použité triedy vozidiel získané z WIM metodiky.

Ako prvý dataset je použitých 1737 vozidiel, ktorý bol augmentovaný na 18396 vozidiel. Z neho je použitých 30% na testovanie. Druhý dataset je zložený z 863 vozidiel. Nebola na ňom vykonaná augmentácia a je použitý len na test.

3.5 Klasifikácia

3.5.1 Vytvorenie neurónovej siete MLP

Z knižnice Tensorflow som využil Keras modely na vytvorenie siete. Na vytvorenie neurónovej siete MLP (multi layer perceptron) som použil sekvenčný typ modelu. Jednoduchým priradením dokážem rozšíriť(pridať vrstvy) do neurónovej siete.

V nasledujúcom výpise bolo ako prvé nutné pridať vstupnú vrstvu, ktorú som zvolil pre vstupný vektor s rozmerom 7. Rozmer je zvolený podľa vstupných dát. Počet náprav nad 8 je ojedinelý a automaticky je zaradený do triedy 13.
Ostatné vrstvy, ktoré som použil:

- Dense(128) - prvá skrytá vrstva so 128 neurónmi a ReLu aktivačnou funkciou
- Dense(64) - druhá skrytá vrstva so 64 neurónmi a ReLu aktivačnou funkciou
- Dropout(0.2) - regularizácia s 20 percentným vypínaním neurónov
- Dense(13) - výstupná vrstva siete s 13 triedami a aktivačnou funkciou softmax

Posledným krokom bola kompilácia sekvenčného modelu. Ako optimalizátor som vybral metódu *Adam* a stratovú funkciu *categorical_crossentropy*.
Optimalizátor *Adam* sa vyznačuje veľkou rýchlosťou. Uchováva váhy z minulosti a upravuje rýchlosť učenia pre každý parameter zvlášť. To znamená, že využíva priemer aj štvorcový priemer gradientov.
Stratová funkcia *categorical_crossentropy* spolu s tzv. One hot encoding je využívaná pri tréningu viactriednej klasifikácie. Čím je pravdepodobnosť správneho zaradenia vzoru do triedy vyššia, tým je strata nižšia. Pomocou funkcie softmax sa vyhodnotia dáta z neurónovej siete a stratová funkcia ich porovná so skutočnosťou.

```
model_perc=Sequential()
model_perc.add(Input(shape=(7,)))
model_perc.add(Dense(128, activation='relu'))
model_perc.add(Dense(64, activation='relu'))
model_perc.add(Dropout(0.2))
model_perc.add(Dense(13, activation='softmax'))
model_perc.compile( optimizer='adam',
                   loss='categorical_crossentropy',
                    metrics=['accuracy'])
model_perc.summary()
```

Tréning modelu

Dataset som rozdelil na tréningové dáta a testovacie dáta v pomere 7 ku 3. Do rozdelenia som zakomponoval aj tzv. Shuffle, teda doslovne zamiešanie datasetu. Týmto krokom som docielil náhodné rozmiestnenie vzoriek, keďže dataset bol spájaný z niekoľkých ďalších menších datasetov. Potrebnou úpravou je aj konvertovanie skutočných výsledkov(label) na one hot encoding.

Posledným krokom pred učením je škálovanie vstupných dát. Pomocou metódy *StandardScaler* som rozmiestnil dáta s rozptylom 1 a s priemerom 0. Týmto krokom som zamedzil vplyv veľkých hodnôt v datasete na celkové učenie sa a zrýchlil konvergenciu siete k správnemu výsledku. To znamená, že nie je potrebné väčšie množstvo tréningových epoch ako je teraz.

Pomocou metódy *.fit* som spustil učenie, ktoré je ukázané v nasledujúcom výpise. Počet epoch som nastavil na 30. Tento počet som zvolil pozorovaním straty a validačnej straty. Ak už ďalej validačná strata neklesá, ale celková strata stále klesá, hrozí preučenie siete. Batch size som zvolil taktiež pozorovaním validačnej straty nájdením optimálnej hodnoty. Validačné rozdelenie je určené na 30 percent datasetu. Vhodným rozdelením by bol aj pomer 8 ku 2 alebo použitie krížovej validácie.

```
model_perc.fit(X_train, y_train,epochs=30,
                batch_size=32, validation_split=0.3)
```

Výsledky

Po natrénovaní neurónovej siete som vytvoril klasifikačnú správu. Je to metóda z knižnice scikit-learn a poskytuje niekoľko metrík, ako napríklad recall, precision alebo F1-score, ktorými vieme zhodnotiť kvalitu siete.

Recall je pomerom správne predikovaných výsledkov ku všetkým výsledkom, ktoré zahŕňajú správne predikované výsledky aj tie, ktoré sieť označila, že nepatria do danej triedy. Týmto spôsobom zistíme, či vie sieť nájsť všetky prípady, ktoré patria danej triede.

Precision je pomerom správne predikovaných pozitívnych výsledkov ku všetkým predikovaným pozitívnym výsledkom. Takto zistíme, ako sieť vie správne určiť, či dané vozidlo patrí do konkrétnej triedy.

Na obrázku č.3.13 je zobrazená konkrétna klasifikačná správa k môjmu modelu. Pri pohľade na jednotlivé triedy je precíznosť veľmi vysoká a to znamená, že klasifikátor dokáže rozoznávať aj triedy, ktoré nemali veľké zastúpenie v datasete. Rozšírením datasetu som dosiahol vyššiu úspešnosť prakticky na všetkých triedach. Recall v triede 13(na obrázku ako trieda 12) je nižší najmä z toho dôvodu, že celkový počet vozidiel tejto triedy bol pred rozšírením datasetu nízky a variabilita pri tejto triede

je veľmi veľká. Nižšia precíznosť pri triede 2 je spôsobená podobnosťou s triedou 1, keďže ide najmä o 2 nápravové vozidlá s podobným rázvorom. Celková presnosť klasifikácie je okolo 97 percent, čo splňuje požiadavky(viac ako 95 percent) na použitie pri pokutovaní vozidiel za mýto.

	precision	recall	f1-score	support
0	0.97	0.86	0.91	444
1	0.78	0.94	0.85	459
2	1.00	0.99	1.00	464
3	1.00	1.00	1.00	454
4	0.99	0.89	0.94	489
5	1.00	1.00	1.00	472
6	1.00	1.00	1.00	453
7	0.88	0.99	0.93	439
8	1.00	1.00	1.00	476
9	1.00	1.00	1.00	492
10	1.00	1.00	1.00	457
11	0.94	1.00	0.97	445
12	1.00	0.83	0.91	435
accuracy			0.96	5979
macro avg	0.97	0.96	0.96	5979
weighted avg	0.97	0.96	0.96	5979

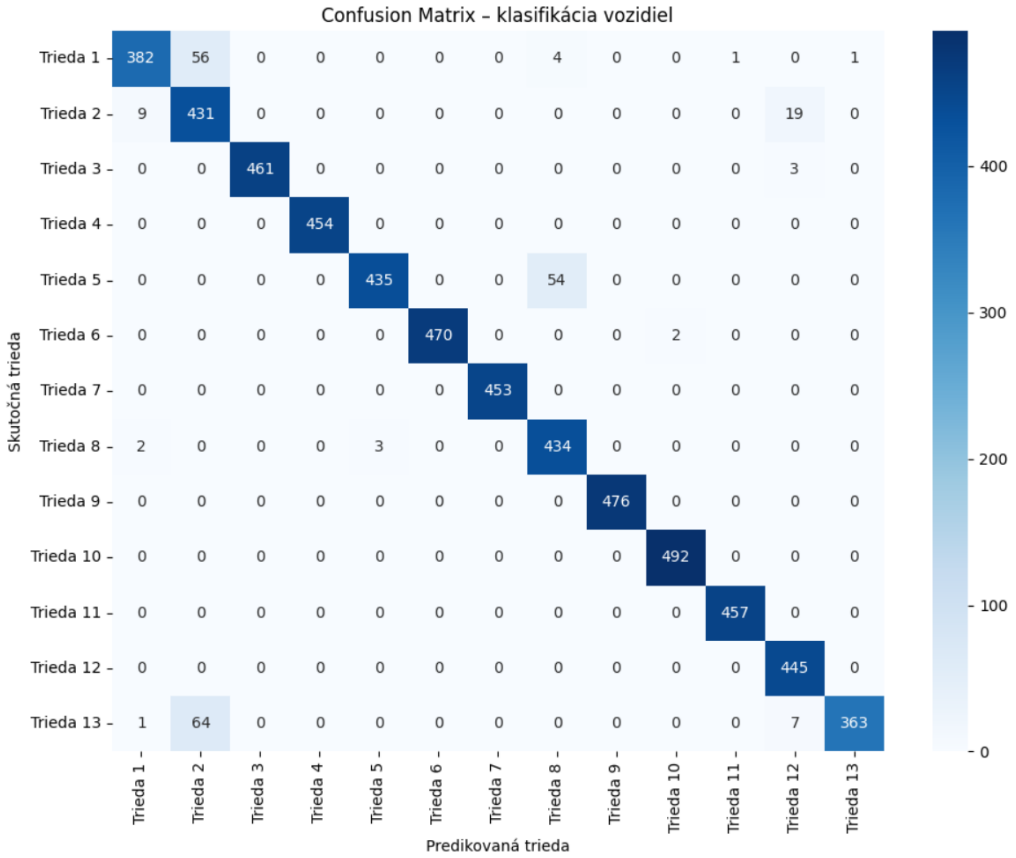
Obr. 3.13: Klasifikačná správa

Na komplexnejšie zobrazenie som využil maticu zámien vytvorenú pomocou kódu v nasledujúcom výpise. Po predikcii výsledkov som spojil výsledky pre každú vzorku, kde som našiel najpravdepodobnejšiu triedu pre každú z nich. Na vytvorenie matice zámien je využitá metóda *confusion_matrix* z knižnice scikit-learn. Na zobrazenie výsledkov som využil knižnicu seaborn s metódou *heatmap*. Parametre, ktoré sa nastavujú v metóde, sú len na zobrazenie a nemajú vplyv na dáta.

```
class_names = [f'Trieda␣{i+1}' for i in range(13)]
y_pred_prob = model_perc.predict(X_test)
y_pred = np.argmax(y_pred_prob, axis=1)
cm = confusion_matrix(np.argmax(y_test, axis=1), y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
        xticklabels=class_names, yticklabels=class_names)
```

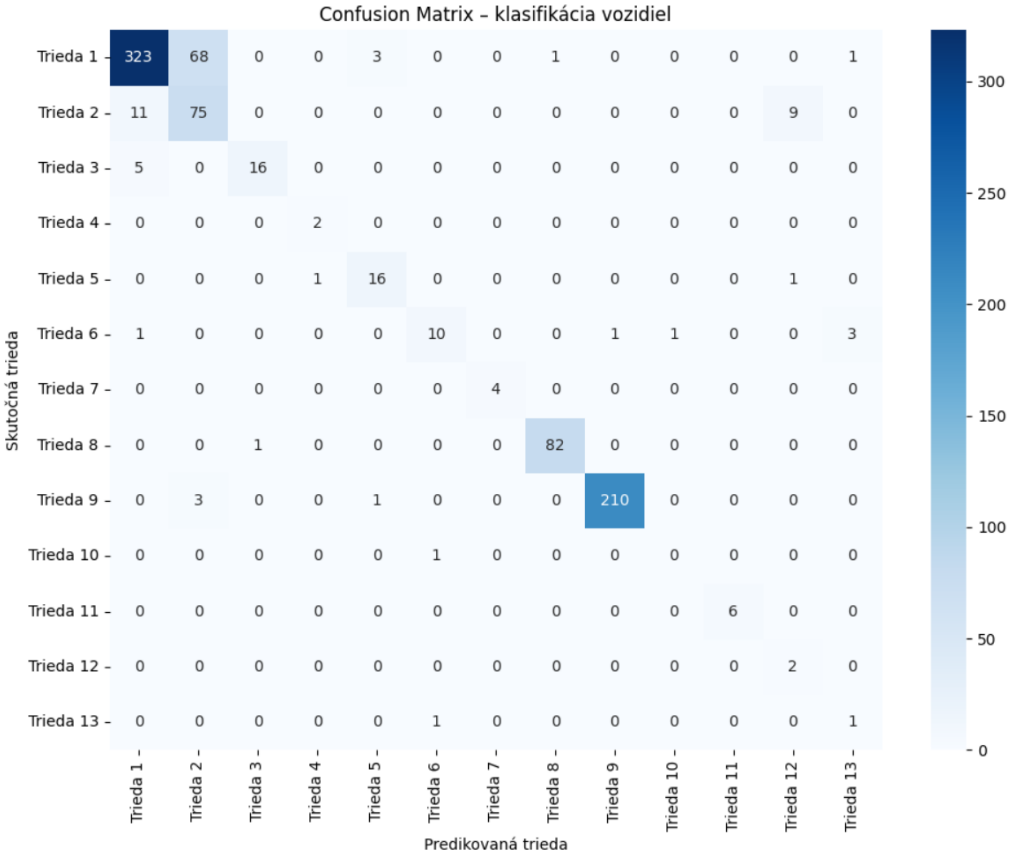
Zobrazením matice zámien sa ukážu triedy, ktoré sa medzi sebou často zamieňajú. V konkrétnom prípade na obrázku č.3.14 ide o triedy 1 a 2, kde dochádza k nesprávnej klasifikácii, ako bolo spomenuté pri klasifikačnej správe. Taktiež sú nesprávne klasifikované vozidlá medzi triedami 2 a 12 a 2 a 13. Po vyhľadaní vozidiel s 2 nápravami v exportovaných WIM súboroch, ktoré sú klasifikované ako trieda 2 a 13 som našiel zhodu v rázvoroch. Podobnosť je daná tým, že do triedy 13 patria traktory, ktoré majú rázvor ako dodávky(malé nákladné vozidlá) v triede 2. Trieda

12 klasifikuje autobusy, ktoré majú tiež podobný rázvor ako dodávky. Pri zámene tried 8 a 5 je možná chyba pri príprave datasetu. Pri pohľade na maticu zámien reálnych dát na obrázku č.3.15 je nesprávna klasifikácia práve v spomínaných triedach. Celková úspešnosť predikcie na týchto dátach je 88 percent.



Obr. 3.14: Matica zámien neurónovej siete s augmentovaným datasetom

Po zohľadnení častých zámien medzi niektorými triedami je tento model použiteľný pre klasifikáciu väčšiny vozidiel. Dôraz na správnu klasifikáciu sa kladie najmä na nákladné vozidlá a pri väčšine tried je úspešnosť nad požadovaných 95 percent.



Obr. 3.15: Matica zámien neurónovej siete s reálnym datasetom

3.5.2 Klasifikácia rozhodovacím stromom

Na klasifikáciu rozhodovacím stromom som využil klasifikátor z knižnice scikit-learn. Vo výpise za odsekom je zobrazená konkrétna metóda rozhodovacieho stromu. Prvý parameter, ktorý som nastavil, je *random_state=42*. Je to inicializačný parameter pre generátor náhodných čísiel a zabezpečuje, že model bude natrénovaný s rovnakým náhodným rozdelením. Číslo 42 nemá žiaden vedecký význam, je to len zaužívaná referencia. Parameter *criterion* je kritérium rozdelenia a zvolil som parameter *gini*, ktorý rýchlejšie konverguje k správnemu výsledku ako metóda informačného zisku (entropia).

```
clf_t = DecisionTreeClassifier(random_state=42,
                               criterion='gini')
```

Na učenie klasifikátora som použil rovnaké dáta ako pri vytvorenej neurónovej sieti v predošlej sekcii. Využil som taktiež augmentáciu datasetu SMOTE technikou.

Výsledky

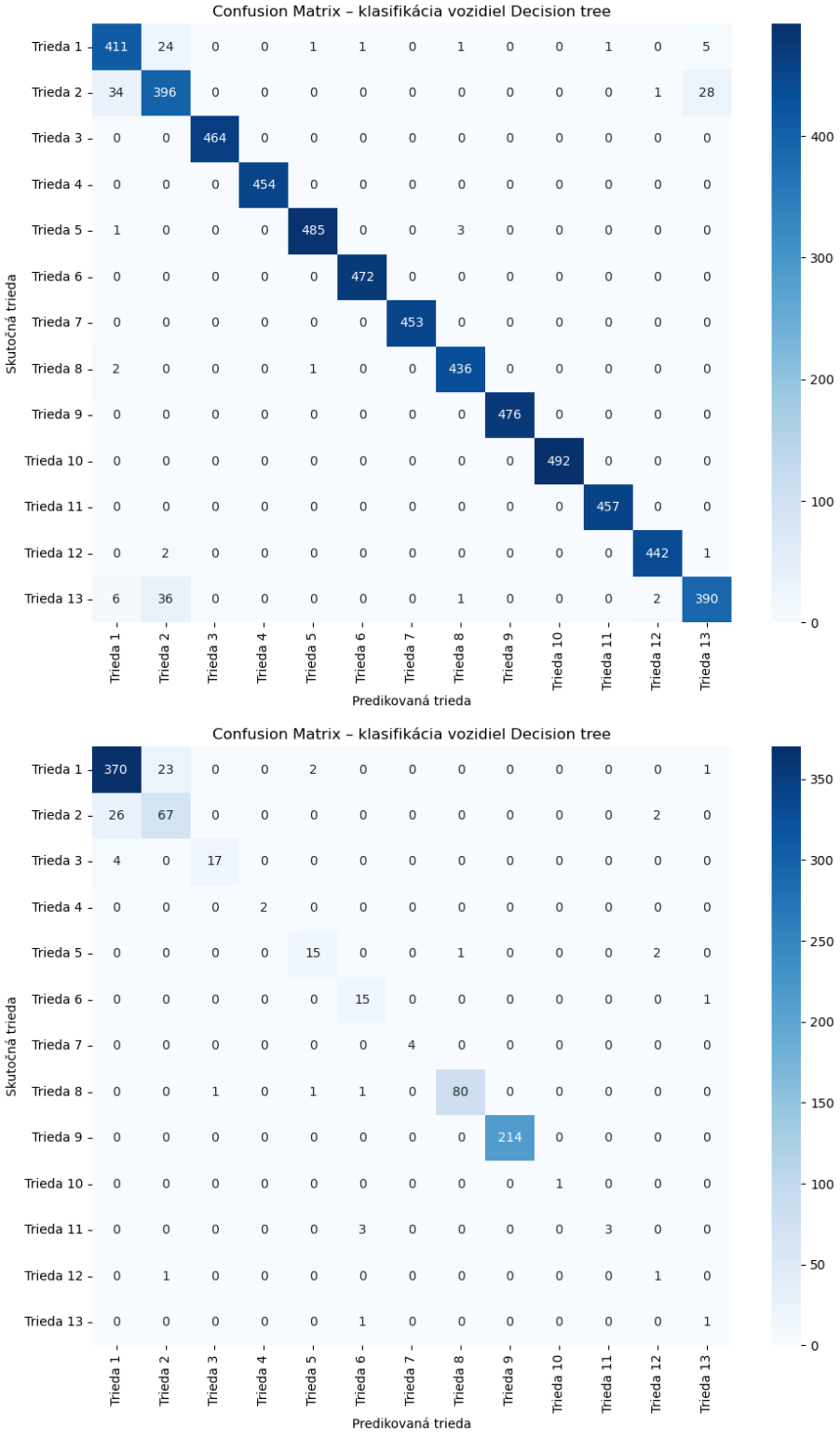
Klasifikátor dosiahol pomocou testu krížovou validáciou vysokú presnosť 97 percent a nulový rozptyl, čo znamená vysokú stabilitu a nízku citlivosť na zmenu dát.

Klasifikačná správa ukazuje zlepšené výsledky precíznosti cca. o 10 percent v druhej triede a markantné zlepšenie v ôsmej triede oproti neurónovej sieti. Negatívom je zhoršená precíznosť v prvej triede, ktorá ale nie je podstatná.

	precision	recall	f1-score	support
1.0	0.91	0.93	0.92	444
2.0	0.86	0.86	0.86	459
3.0	1.00	1.00	1.00	464
4.0	1.00	1.00	1.00	454
5.0	1.00	0.99	0.99	489
6.0	1.00	1.00	1.00	472
7.0	1.00	1.00	1.00	453
8.0	0.99	0.99	0.99	439
9.0	1.00	1.00	1.00	476
10.0	1.00	1.00	1.00	492
11.0	1.00	1.00	1.00	457
12.0	0.99	0.99	0.99	445
13.0	0.92	0.90	0.91	435
accuracy			0.97	5979
macro avg	0.97	0.97	0.97	5979
weighted avg	0.97	0.97	0.97	5979

Obr. 3.16: Klasifikačná správa rozhodovacieho stromu

V matici zámien na obrázku č.3.17 je vidieť podobný problém ako pri neurónovej sieti, a to nesprávnu klasifikáciu medzi triedami 1 a 2 a medzi triedami 2 a 13. Pohľadom na maticu zámien je vidieť zlepšený recall. Matica zámien s reálnymi dátami na obrázku zobrazuje podobnú situáciu, ktorú ale ťažšie zhodnotiť pre triedy 10 až 13, keďže obsahujú málo vzoriek.



Obr. 3.17: Matice zámien rozhodovacieho stromu s augmentovaným datasetom a reálnym datasetom

3.5.3 Klasifikácia logistickou regresiou

V logistickej regresii som nastavoval parameter *solver* a parameter *max_iter*. Ako optimalizačnú metódu som zvolil IBFGS algoritmus, ktorý je vhodný na viactriednu klasifikáciu a pri menšom počte dát. Počet iterácií som zvolil po skúšaní niekoľkých hodnôt. Zvolil som tú najnižšiu, pri ktorej neklesala celková úspešnosť klasifikácie.

```
clf_lr=LogisticRegression(random_state=42,
                solver='lbfgs',max_iter=500)
```

Výsledky

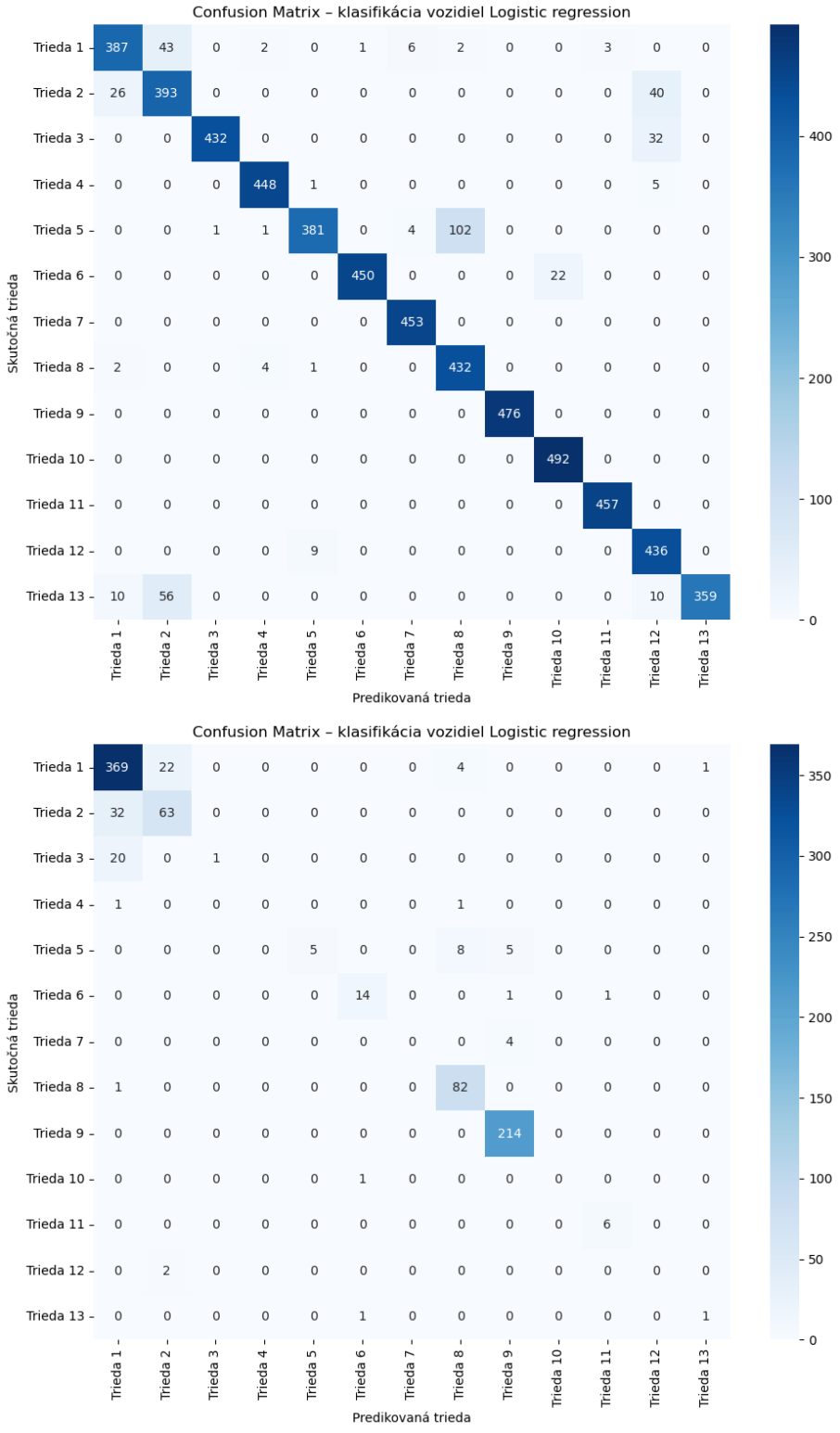
Výsledok testu pomocou metódy krížovej validácie je vysoká presnosť, konkrétne 93 percent, a taktiež nulový rozptyl.

Klasifikátor dosahuje zníženú precíznosť pri triede 12, 8 alebo 2 oproti rozhodovaciemu stromu. Takisto je vidieť menej zachytených pravých pozitívnych pre väčšinu tried.

	precision	recall	f1-score	support
1.0	0.91	0.87	0.89	444
2.0	0.80	0.86	0.83	459
3.0	1.00	0.93	0.96	464
4.0	0.98	0.99	0.99	454
5.0	0.97	0.78	0.86	489
6.0	1.00	0.95	0.98	472
7.0	0.98	1.00	0.99	453
8.0	0.81	0.98	0.89	439
9.0	1.00	1.00	1.00	476
10.0	0.96	1.00	0.98	492
11.0	0.99	1.00	1.00	457
12.0	0.83	0.98	0.90	445
13.0	1.00	0.83	0.90	435
accuracy			0.94	5979
macro avg	0.94	0.94	0.94	5979
weighted avg	0.94	0.94	0.94	5979

Obr. 3.18: Klasifikačná správa logistickej regresie

Z prvej matice zámien na obrázku č.3.19 je očividná zámena tried 5 a 8. Tieto 2 triedy majú rozdielne rozloženie náprav a jedinou spoločnou vlastnosťou je možný počet náprav. Ďalej dochádza k zámene tried 2, 12 a 13, čo je aktuálne pri všetkých mnou vytvorených klasifikátoroch. Matica s reálnymi dátami vykazuje novú zámenu a to medzi triedami 3 a 1. Možnou príčinou je podobnosť medzi osobným autom s prívesom s dvoma nápravami a menším nákladným vozidlom s troma nápravami.



Obr. 3.19: Matice zámien logistickej regresie s augmentovaným datasetom a reálnym datasetom

3.5.4 Klasifikácia support vector machine

Klasifikátor support vector machine som nastavil na lineárny klasifikátor, to znamená, že má priamky ako rozhodovacie hranice. Lineárny klasifikátor som zvolil z dôvodu porovnania s logistickou regresiou, keď majú klasifikovať skoro lineárne separovateľné dáta, ktoré sú v datasete. Pred vstupom do klasifikátora boli dáta škálované kvôli veľkej citlivosti algoritmu na zmeny vstupu.

```
clf_svm=svm.LinearSVC(random_state=42)
```

Výsledky

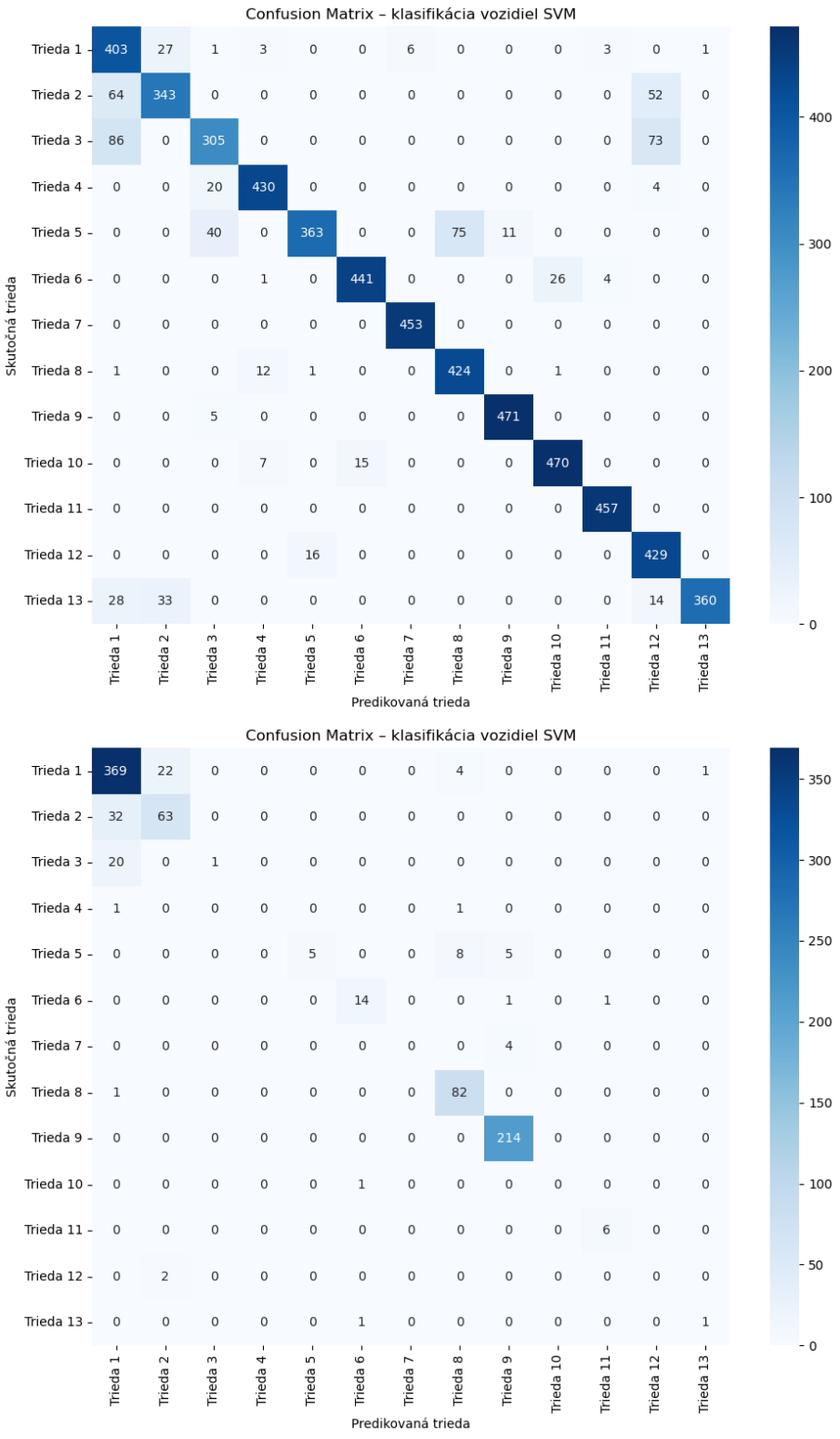
Uvedený klasifikátor dosiahol najnižšiu úspešnosť spomedzi všetkých realizovaných klasifikátorov, a síce 90-percentnú presnosť a taktiež nulový rozptyl.

Recall pri triedach 2, 3 a 5 je nižší oproti ostatným klasifikátorom.

	precision	recall	f1-score	support
1.0	0.69	0.91	0.79	444
2.0	0.85	0.75	0.80	459
3.0	0.82	0.66	0.73	464
4.0	0.95	0.95	0.95	454
5.0	0.96	0.74	0.84	489
6.0	0.97	0.93	0.95	472
7.0	0.99	1.00	0.99	453
8.0	0.85	0.97	0.90	439
9.0	0.98	0.99	0.98	476
10.0	0.95	0.96	0.95	492
11.0	0.98	1.00	0.99	457
12.0	0.75	0.96	0.84	445
13.0	1.00	0.83	0.90	435
accuracy			0.89	5979
macro avg	0.90	0.90	0.89	5979
weighted avg	0.90	0.89	0.89	5979

Obr. 3.20: Klasifikačná správa SVM klasifikátora

Z matice zámien na obrázku č.3.21 je jasne vidieť nižší recall pri triedach 3 a 5. Pri testovaní s reálnymi dátami dochádza k rovnakej situácii.



Obr. 3.21: Matice zámien SVM klasifikátora s augmentovaným datasetom a reálnym datasetom

Záver

V bakalárskej práci som sa venoval použitiu piezoelektrických senzorov na klasifikáciu vozidiel v premávke. V prvej kapitole som predstavil základné princípy zberu dát pomocou piezoelektrických senzorov a priblížil som funkčný princíp piezoelektrických čidiel. V druhej kapitole sú ukázané základné metódy oddelenia a klasifikácie dát pomocou viacerých algoritmov.

Zhlukovaním a metódou hustoty jadra som v kapitole 3 oddelil dáta patriace jednému vozidlu. Na realizáciu niektorých algoritmov ako napr. DBSCAN algoritmus alebo ďalšie som použil knižnicu Sci-kit a k tomu som naprogramoval vlastný kód. Parametre potrebné na presné oddelenie vozidiel som určil pomocou štatistických metód alebo jednoduchým odhadovaním parametra s korekciou vopred daných kontrolných dát. Ďalším programom, ktorý som vyskúšal, bol konvenčný prístup k separácii vozidiel pomocou priameho výpočtu rýchlosti. Program som ale nevedel dobre odladiť na datasetoch, kde sa vyskytoval šum alebo nespojené nápravy. Výsledkovo najlepšie vyšiel mnou realizovaný algoritmus DBSCAN. Počet vozidiel pri porovnaní s referenčnou klasifikáciou WIM je prakticky rovnaký.

Na spracovanie dát som využil vlastné algoritmy a následne som vypočítal rýchlosti vozidiel a vzdialenosti náprav. Algoritmus má ale problémy pri spracovaní vozidiel, ktoré idú tesne za sebou a nespracováva rôzne smery jazdy.

Ako pilotné labelovanie som realizoval porovnanie s tabuľkou EUR13 obsahujúcu rozsahy náprav. Následne kvôli vyššej spoľahlivosti som zvolil labely z WIM databázy. Poslednými krokmi bola úprava veľkosti datasetu, augmentácia a padding.

Na klasifikáciu som použil algoritmy z knižnice Sci-kit a Tensorflow keras. Najvyššiu presnosť som dosiahol pri MLP neurónovej sieti a to 97%. Najväčšiu presnosť pri modeloch scikit-learn som dosiahol s algoritmom rozhodovací strom a to 97% a najnižšiu pomocou SVM algoritmu, ktorý má presnosť 90%. Hodnota pod 95% je na použitie napr. na kontrolu mýta nie veľmi vhodná. Problémom je nízka variancia datasetu kvôli zloženiu dopravy naklonenej prirodzene k osobným autám.

Cieľom do budúcna by malo byť určite pokrytie všetkých problematických scenárov. Tie môj algoritmus nerozoznáva veľmi dobre. Vzhľadom na okolnosti mi ale táto práca priniesla cenné poznatky o algoritmoch strojového učenia, dozvedel som sa veľa nových informácií ohľadom ladenia hyperparametrov, augmentácie datasetu, neurónových sietí a zároveň som si zlepšil svoje programátorské schopnosti.

Literatúra

[1] *Traffic Monitoring Guide.* Online. Washington, D.C., 2024. Dostupné z: https://www.fhwa.dot.gov/policyinformation/tmguide/2022_TMG_Updated_20241008.pdf. [cit. 2024-11-26].

[2] ĎAĎO, Stanislav a KREIDL, Marcel. *Senzory a měřicí obvody.* Praha: České vysoké učení technické, 1996. ISBN 80-01-01500-9.

[3] HALLENBECK, Mark a WEINBLATT, Herbert. *Equipment for Collecting Traffic Load Data.* Online. Transportation Research Board, 2004. Dostupné z: https://trb.org/publications/nchrp/nchrp_rpt_509.pdf. [cit. 2024-11-18].

[4] RASCHKA, Sebastian; LIU, Yuxi a MIRJALILI, Vahid. *Machine learning with PyTorch and scikit-learn: develop machine learning and deep learning models with Python* Birmingham: Packt, 2022. ISBN 978-1-80181-931-2.

[5] JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert a TAYLOR, Jonathan E. *An introduction to statistical learning: with applications in Python.* Springer texts in statistics. Cham: Springer, 2023. ISBN 978-3-031-39189-7.

[6] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras & TensorFlow: concepts, tools, and techniques to build intelligent systems.* 3rd ed. Sebastopol, CA: O'Reilly Media, 2023. ISBN 978-1-098-12597-4..

[7] *Jednoduchá lineárna regresia. Pearsonov výberový korelačný koeficient.* Online. 2013. Dostupné z: https://kurzy.kpi.fei.tuke.sk/nm/student/13.html. [cit. 2024-12-12].

[8] *Maximum likelihood estimation.* Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation,2001-. Dostupné také z: https://en.wikipedia.org/wiki/Maximum_likelihood_estimation. [cit. 2024-12-12].

[9] *Sigmoid function explained in less than 5 minutes.* Online. In: Medium. 2020. Dostupné z: https://medium.com/@gabriel.mayers/sigmoid-function-explained-in-less-than-5-minutes-ca156eb3049a. [cit. 2024-12-12].

[10] SAFFAR, Or Herman. *An Approach for Choosing Number of Clusters for K-Means.* Online. In: Medium. 2021. Dostupné z: https://towardsdatascience.com/an-approach-for-choosing-number-of-clusters-for-k-means-c28e614ecb2c. [cit. 2024-12-27].

[11] KULESHOV, Volodymyr. *Lecture 9: Density Estimation.* Online. Applied Machine Learning. 2023. Dostupné z: https://kuleshov-group.github.io/aml-book/contents/lecture9-density-estimation.html. [cit. 2024-12-30].

[12] *Kernel density estimation.* Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation,2001-. Dostupné také z: https://en.wikipedia.org/wiki/Kernel_density_estimation#cite_note-SI1998-22. [cit. 2024-12-30].

[13] GOODFELLOW, Ian; BENGIO, Yoshua a COURVILLE, Aaron. *Deep learning.* Cambridge, Massachusetts: The MIT Press, 2016. ISBN 978-0-262-03561-3.

[14] BISHOP, Christopher M. *Pattern recognition and machine learning.* New York: Springer, 2006. ISBN 978-0-387-31073-2.

[15] *Multilayer Perceptrons in Machine Learning: A Comprehensive Guide.* Online. 2025. Dostupné z: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning. [cit. 2025-05-01].

[16] BUDUMA, Nikhil. *Fundamentals of deep learning: designing next-generation machine intelligence algorithms.* Beijing: O'Reilly, 2017. ISBN 978-1-4919-2561-4.

[17] *Support Vector Machine vs. Random Forest for Remote Sensing Image Classification: A Meta-analysis and systematic review.* Online. In: Research Gate. 2020. Dostupné z: //www.researchgate.net/figure/An-SVM-example-for-non-linearly-separable-data-with-the-kernel-trick_fig3_344437400. [cit. 2025-05-20].

Zoznam príloh

A Obsah elektronickej prílohy

- `/` ..... koreňový adresár
  - `dataset` ..... všetky potrebné datasety
    - `20241113T0800014120100_AxleEventRecords.csv` ..... dataset
    - `20250106T0800002200100_AxleEventRecords.csv` ..... dataset
    - `20250106T0900041270100_AxleEventRecords.csv` ..... dataset
    - `20250107T0800034230100_AxleEventRecords.csv` ..... dataset
    - `20250107T0900265230100_AxleEventRecords.csv` ..... dataset
    - `complet_eur13.csv` ..... dataset separovaných vozidiel
    - `complet_eur13_2024_allaxcmp.csv` ..... dataset separovaných vozidiel
    - `complet_eur13_2025_allaxcmp6.csv` ..... dataset separovaných vozidiel
    - `complet_eur13_2025_test1.csv` ..... dataset separovaných vozidiel
    - `complet_eur13_2025_test2.csv` ..... dataset separovaných vozidiel
    - `wimclass.csv` ..... prevod WIM na EUR13
    - `WimVehicles_20241113T1026145690100.csv` ..... WIM databáza
    - `WimVehicles_20250106T0812484820100.csv` ..... WIM databáza
    - `WimVehicles_20250107T0803463370100.csv` ..... WIM databáza
  - `models` ..... Uložené modely klasifikátorov
    - `decisiontree.joblib` ..... model rozhodovací strom
    - `linsvm.joblib` ..... model lineárnej SVM
    - `logisreg.joblib` ..... model logistickej regresie
    - `mlp.keras` ..... model MLP
  - `clust_scikit_kmeans.py` ..... program K means v pythone
  - `conv_meth.m` ..... konvenčná metóda spracovania náprav
  - `final_mlp.ipynb` ..... program siete multilayer perceptron v pythone
  - `kernel_density_est.py` ..... program na odhad hustoty jadra v pythone
  - `label_class.ipynb` ..... program kompletnej klasifikácie v pythone
  - `readme.md` ..... návod
  - `scikit_dbscan.py` ..... program DBSCAN v pythone