Szakdolgozat

Miskolci Egyetem

Szakdoga címe

Készítette:

Ferencsik Dorina Kinga

Évfolyam. szak-szak

Témavezető:

Egyik konzulens neve

Másik konzulens neve...

Miskolc, 2019

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematika Tanszék **Szám:**

Szakdolgozat Feladat

Ferencsik Dorina Kinga (YXI8DJ) programtervező informatikus jelölt részére

A szakdolgozat tárgyköre: numerikus eljárások tesztelése

A szakdolgozat címe:

A feladat részletezése:
Nemlineáris egyenletrendszer megoldására szolgáló ABS-módszerek áttekintő jellegű ismertetése.

Témavezető(k): Dr. (vagy nem doktor) Valamilyen Valaki beosztás (pl. egyetemi adjunktus)

Konzulens(ek): (akkor kötelező, ha a témavezető nem valamelyik matematikai tanszékről való; de persze lehet egyébként is)

A feladat kiadásának ideje:

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott .....; Neptun-kód: .....
a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős .....
szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy .....
című szakdolgozatom/diplomatervem saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ...év ..... .hó ..... .nap

.....
Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. Bevezetés **6**

2. Téma elméleti kifejtése **8**
2.1. Mobil alkalmazások . . . 8
2.1.1. Bike calculator . . . 8
2.1.2. Strava . . . 8
2.1.3. Endomondo . . . 9
2.1.4. Best Bike Split . . . 9
2.1.5. Összefoglalás és célkitűzés . . . 9
2.2. Megválaszolandó kérdések . . . 11
2.2.1. Nehézségi osztályok meghatározása - eddigiek . . . 11
2.2.2. Nehézség meghatározása - új útvonalra . . . 12
2.2.3. Időtartam becslés . . . 12
2.2.4. Edzésterv . . . 12
2.3. Gépi tanulás . . . 13
2.3.1. Regresszió . . . 13
2.3.2. Osztályozás . . . 14
2.3.3. Klaszterezés . . . 14
2.4. A gépi tanulás alapköve - adatok . . . 15
2.4.1. Adatgyűjtés . . . 15
2.4.2. Adatstruktúra . . . 16

3. Fejlesztői dokumentáció **18**
3.1. Adathalmaz előkészítés . . . 18
3.1.1. Adatstruktúra kialakítása . . . 18
3.1.2. Adattisztítás . . . 19
3.1.3. Adatok áttekintése . . . 22
3.1.4. További adat feldolgozás . . . 24
3.2. Gépi tanulás . . . 27
3.2.1. Mozgási és eltelt idő számítása . . . 27

4. Összefoglalás **29**

Irodalomjegyzék **30**

Adathordozó használati útmutató **31**

1. fejezet

Bevezetés

Először 2018. nyarán kezdtem el az adatbányászattal és gépi tanulással foglalkozni és ahogy mind többet értettem meg belőle inkább érdekelt a terület, a kihívásai. A lelkesedés kitartott így ősszel a témaválasztás határidejének közeledtével az lebegett a szemem előtt hogy a területhez szorosan kapcsolódó szakdolgozatot tudjak elkezdeni.

Az irány megszabása után egy konkrét célt kellett találni, egy alkalmazási területet ahol a gépi tanulás jól hasznosítható, olyan problémák, feladatok vannak amihez érdemes lehet gépi tanulás algoritmusokat használni. Hosszas mérlegelés után a különböző sportok közül a kerékpározásra esett a választás. Ennek több oka is volt. Elsősorban a kerékpározás népszerű, elterjedt mozgásforma, így esélyesnek tűnt hogy viszonylag nagy mennyiségű statisztikai adatot lehet róla gyűjteni, változatos sportolói körből. Továbbá mikor időm engedi én is kerékpározok és ezeket az utakat rögzítem, így néhány kiinduló adat már elérhető volt ami alapján el tudtam kezdeni a tervezést. Az utak nyomon követésére a Strava mobil és webes alkalmazás szolgált, ami részletesen tárolja az egyes útvonalak statisztikai adatait. Az elérhető adatokat elemezve behatároltam több feladatot mint pl.:

- Egy adott útvonal jellemzőinek ismeretében a teljesítéshez szükséges várható időtartam meghatározása. Ez az időtartam két részre választható szét, megkülönböztetjük a mozgással töltött időt a teljes eltelt időtartamtól ami a mozgási időből és a járulékos időveszteségekből adódik mint a forgalmi lámpánál várakozás, pihenés, hosszabb utak esetén étkezés.
- Már megtett utak csoportosítása, nehézségi fokozatok meghatározása.
- A nehézségi szintek alapján edzéstervek készítése.

Azonban a sportteljesítmény becslése nehéz feladat, mivel egyénenként nagyon eltérő, valamint az út nyers adatain kívül az egyén pillanatnyi fizikai és szellemi állapota is nagyban befolyásolja a teljesítményét (álmosság, kimerültség, kedvtelenség stb.). Ezért el kell dönteni hogy egy általános modell megépítése a cél, ami esetleg életkor, nem, edzettségi szint szerint finomhangolható azonban kisebb pontossággal rendelkezik, vagy személyre szabott, pontos becsléseket kell elérni ahol viszont a gépi tanulás során nehézséget okozhat a kis méretű adathalmaz, hiszen kevesen rendelkeznek akár csak egy-két ezer rögzített úttal, míg a megfelelő pontossághoz az egyes algoritmusoknak ennél nagyságrendekkel több adatra van szüksége.

Az elsődleges cél általános modellek építése, ahol egy feladat megoldásához különböző gépi tanulási algoritmusokat lehet alkalmazni, ezeket egyenként javítani, ma-

ximalizálni a pontosságukat. Utána pedig az algoritmusokat össze kell hasonlítani a teljesítményük, felhasznált módszertanaik alapján.

2. fejezet

Téma elméleti kifejtése

2.1. Mobil alkalmazások

Napjainkban rengeteg különböző kerékpáros alkalmazás érhető el amik más más területre fókuszálnak. A következő pontokban ezekre láthatunk néhány példát az egyszerű mérnöki alapú számítástól az útvonalak nyomon követésén keresztül a fejlett, összetett tervező rendszerekig. Látható hogy a felhasználók körében nagy népszerűségnek örvend a különböző alkalmazások közösségi funkciói ahol lehetőség van az ismerősök megtalálására, útvonalak megosztására.

2.1.1. Bike calculator

A "Bike Calculator" [1] weboldal és mobil alkalmazás biciklis becsléseket kínál a felhasználói számára, teljesen mérnöki szempontból. A modell figyelembe veszi a sebesség, erőkifejtés, szélellenállás, gravitáció és egyéb tényezők összefüggését azonban egy általános eredményt ad, nincs lehetőség a felhasználó egyéni teljesítményének figyelembe vételére. Ezzel együtt is egy érdekes megoldás ami rengeteg paramétere révén jól skálázható.

2.1.2. Strava

A Strava [2] az egyik legelterjedtebb alkalmazás a sporttevékenységüket nyomon követni kívánók körében, széles funkcionalitással és hatalmas felhasználói bázissal. Lehetőséget nyújt többek között biciklizés, futás, úszás, hegymászás és rengeteg egyéb sportág eredményeinek rögzítésére és korlátozott szintű tervezésére. A tevékenység során rögzített adatokból alapvető statisztikákat készít a felhasználók számára, mint a sebesség és a tengerszint feletti magasság változása valamint a sportoló pulzusának mozgása. Népszerűsíti a Strava-t hogy lehetőséget kínál az ismerősök megtalálására és az útvonalak, tevékenységek megosztására valamint kommentek és "kudo"-k egyfajta elismerés, tetszés nyilvánítás adására. Érdekes funkciói közé tartozik a szegmensek, rövid útvonal szakaszok kezelése ahol a felhasználóknak lehetőségük van rövid távon versenyezni, előrébb kerülni a szegmenst teljesítők listáján valamint számon tartani a saját rekordjaikat.

A Strava tervezője, jövővel kapcsolatban segítő funkciója egy útvonal tervezőben merül ki,ami segíti a felhasználókat a legnépszerűbb út megtalálásában A és B pont között, térképpel és részletes navigációs leírásokkal. Az útvonal kijelölésénél az alapvető

adatokon kívül egy becslést is kapunk a várható időre nézve. A várható idő számolása a bejelentkezett felhasználó legutóbbi 4 heti teljesítménye alapján történik. Előrelépés hogy egyáltalán figyelembe veszi a felhasználó eddigi útvonalait azonban ennek a módszernek is vannak hiányosságai így csak egy közepes kiindulási alapot ad. Alapvető gond ezzel a becsléssel hogy az útvonalak rögzítésénél a felhasználó hiába állítja be az út típusát (verseny, edzés, ingázás stb.) a Strava számítás során ezeket nem veszi figyelembe, így könnyen előfordulhat hogy a felhasználó kiugróan pontatlan, akár használhatatlan eredményt kap. Például az előző hetekben a saját határait feszegetve versenyre készült, most viszont szeretne egy nyugodt családi biciklizésre elindulni - ekkor nyilván nem fog rekordokat dönteni és a Strava által számolt várható idő semmilyen tényleges információval nem fog szolgálni neki.

2.1.3. Endomondo

Az Endomondo [3] webes és mobil alkalmazás sporttevékenység nyomon követését hivatott megkönnyíteni a felhasználói számára. Rengeteg különböző sportágat megkülönböztet, ezekről külön statisztikákat készít valamint egyéni célokat lehet ezekre vonatkozóan beállítani. Az egyes sportágakat GPS alapú jelkövetés segítségével lehet nyomon követni, illetve kézzel is létrehozhatóak új tevékenységek, továbbá lehetőség van különböző alkalmazásokból vagy fájlból importálni korábban teljesített útvonalakat. Az alkalmazás megvalósít közösségi funkciókat is, a felhasználóknak lehetőségük van megkeresni az ismerőseiket akikkel később megoszthatják legújabb sporttevékenységüket valamint versenghetnek egymással a közösen felállított célok eléréséért. Az Endomondo érdekes adaléka hogy mozgás közben audió visszajelzést biztosít a felhasználók számára az addigi teljesítményükről, így a felhasználók sportolás közben is pontos információval rendelkezhetnek anélkül hogy az alkalmazást futtató eszközüket kellene figyelniük.

2.1.4. Best Bike Split

A Best Bike Split [4] egy rendkívül összetett rendszer amely elsősorban versenyzők számára kínál széles funkcionalitást. Alapköve a versenytáv megtételének optimalizálása melynek keretei között személyre szabott, szakaszokra bontott tervet állít fel a felhasználó számára annak érdekében hogy a tervezett távot a lehető legjobb idő alatt teljesítse. Ez a terv alapul veszi a felhasználó biciklijének és felszerelésének a tulajdonságait (kerekek típusa, bicikli súlya, sisak kialakítása stb.) a preferált versenyzési módot (elhelyezkedés a verseny során, sík terep illetve emelkedő esetén) valamint már teljesített edzést, versenyeket és természetesen a teljesítendő verseny paramétereit. Prémium előfizetés használata során a kalkulációt befolyásolja a várható időjárás is. A rendszer használható tetszőleges útvonalakra azonban az átlag kerékpáros számára szükségtelenül bonyolult, összetett így elsősorban versenyzők számára hasznos.

2.1.5. Összefoglalás és célkitűzés

Több tucatnyi hasonló alkalmazás létezik, webes felülettel illetve Android és iOS lefedéssel hiszen a kerékpározás fellendülőben van napjainkban, így egyre több alkalmazás készül a felhasználói igények kielégítésére mind versenyzők mind pedig hétköznapi felhasználók számára akiknek a biciklizés inkább közlekedési mint sport eszköz. Ezen

alkalmazások felsorolása vagy részletezése nem célja ennek a szakdolgozatnak, azonban néhány példa alapján is könnyen meghatározhatóak általános csoportok amelyekbe a létező megoldások besorolhatóak.

TODO táblázat a biciklis alkalmazásokról (összefoglalás hogy mi mit tud)

Útvonal rögzítés

A legalapvetőbb funkció amire különböző megoldások léteznek. A legegyszerűbb eszközök esetén például csak az aktuális sebesség és a távolság kerül megállapításra egy, a kerékre rögzíthető érzékelőnek a segítségével. Az érzékelő használatával számolni lehet hogy a kerék hányszor fordul körbe adott időablakon belül, ennek és a kerék méretének felhasználásával könnyen megállapítható a pillanatnyi sebesség illetve az útvonal kezdete óta megtett távolság. Azonban a növekvő igényekhez igazodva ma már lényegesen elterjedtebbek a GPS alapú mobil alkalmazások amik a pillanatnyi sebességen és távolságon kívül pontosan tudják rögzíteni az út egyéb paramétereit is mint pl.: kezdő és végpont koordinátái, mozgással töltött idő, megtett emelkedő, legalacsonyabb és legmagasabb pont, átlag sebesség, maximum sebesség. Ezek a jellemzők lényegesen pontosabb leírást adnak egy-egy útvonalról, letárolásukkal könnyen nyomon követhető a személyes fejlődés, kitűzhetőek heti, havi, éves célok. Az útvonal paramétereinek nyomon követése, struktúrája és tárolása megvalósításonként változó, azonban a cél mindig ugyanaz: minél több jellemző meghatározása a lehető legrészletesebb módon.

Közösségi funkciók

A kerékpáros útvonalakat nyomon követő mobil alkalmazások elterjedésével egy új tendencia jelent meg; egyre többször figyelhető meg valamilyen jellegű közösségi funkció megvalósítása az alkalmazáson belül, amik nagy népszerűségnek örvendenek az alkalmazást használók körében. Ezen funkciók nagyban különbözhetnek, általánosságban lehetőség van az alkalmazást használó ismerősök megtalálására, megtett útvonalak megosztására illetve valamilyen formában tetszésnyilvánítás / elismerés adására. Ezeken kívül általában megjelenik valamilyen ösztönzés, lelkesítés ami segíthet a teljesítmény javításában, a kitűzött célok elérésében. A Strava esetében ez a szegmensek formájában jelenik meg, amik lényegében rövid, maximum néhány száz méteres útvonalak, általában valamilyen szempontból nehezebb szakaszok (pl.: emelkedő). Ezen szegmensek teljesítésével a felhasználó automatikusan felkerül több rangsorra is (globális, erre az évre vonatkozó, nem és korcsoport alapján lebontott) ahol lehetősége van ismerősökkel és ismeretlenekkel vetélkedni ezzel felébresztve a versenyszellemet.

Tervezés, becslés

A kerékpáros alkalmazások fejlődésének következő lépcsőfoka az útvonalak tervezése és a teljesítmény becslése. Útvonalak tervezésére kevésbé kerékpár specifikus alkalmazások is lehetőség adnak mint például a Google Térkép de sok, kifejezetten kerékpárosoknak készült alkalmazásban is léteznek megvalósítások. A tervezés különböző szempontok alapján történhet: egyik fontos szempont a távolság (ez a legfontosabb amennyiben a cél az út minél gyorsabb megtétele A és B pont között), egy másik pedig a népszerűség szóval foglalható össze. Az utóbbi főleg akkor fontos ha a kerékpározásra nem közlekedés szerűen tekintünk hanem inkább mint sport, kikapcsolódás hiszen ilyenkor előrébb kerül a jogos felhasználói igény az olyan útvonalakra ahol szép környezetben,

lehetőség szerint autós és gyalogos forgalomtól zavartalanul lehet biciklizni. Az útvonalak tervezéséhez szorosan kapcsolódik a teljesítmény, a várható idő meghatározása ami koránt sem egyszerű feladat. Ennek legalapvetőbb megoldása a Bike calculator által is megvalósított mérnöki, fizikai modell, ahol a teljesítmény tisztán a felszerelés és a megtenni kívánt út paraméterei alapján kerül kiszámításra

Tervezés, becslés - egyéni teljesítmény alapján

Az útvonal tervezés és a várható teljesítmény becslésének továbbfejlesztésének, pontosításának alapjául az eddig megtett útvonalak és az elért eredmények szolgálhatnának, ez azonban egy olyan terület ahol még nem sok megoldás született. Ide sorolható a Strava által fejlesztett útvonal tervező, ahol a várható idő becsléséhez alapul veszi a bejelentkezett felhasználó elmúlt 4 heti teljesítményét, és míg ez egy jó kezdeményezés a valóságban nem mindig szolgál pontosabb információval mint egy egyszerű, fizikai összefüggéseken alapuló általános modell. Megnehezíti a pontos becslések készítését hogy egy adott felhasználó sokszor kevés adattal rendelkezik amelyek nem elég változatosak egy új útvonal esetén várható teljesítmény megbecsléséhez.

Ezen 4 fő csoport közül az első három által meghatározott problémákra, igényekre mára már rengeteg, változatos módon megvalósított megoldás létezik, így ezeknek a tárgyalásától a továbbiakban eltekintünk. Azonban az utolsó probléma, a felhasználó egyéni teljesítményének figyelembe vétele egy nyitott terület, ahova nem sokan merészkedtek még. **TODO** ez a szakdoga célja

minden sport esetén az adott ember, sportoló az alapvető tényező, az ő képességei fogják meghatározni legnagyobb részben az eredményeit. Természetesen a felszerelés és a terepviszonyok is nagy hatással vannak a végeredményre de az emberi tényező nem hagyható ki a számítások során amennyiben növelni akarjuk a pontosságot.

2.2. Megválaszolandó kérdések

A sportteljesítmény becslése nehéz feladat, mivel egyénenként nagyon eltérő lehet, valamint az út nyers adatain kívül (távolság, emelkedő, időjárás) a egyén pillanatnyi fizikai és szellemi állapota is nagyban befolyásolja (álmosság, kimerültség, kedvtelenség stb.). Ezért el kell dönteni hogy egy általános modell megépítése a cél, ami esetleg életkor, nem, edzettségi szint szerint finomhangolható azonban kisebb pontossággal rendelkezik, vagy személyre szabott, pontos becsléseket kell elérni ahol viszont a gépi tanulás során nehézséget okozhat a kis méretű adathalmaz, hiszen kevesen rendelkeznek akár csak néhány száz rögzített úttal, míg a megfelelő pontossághoz ehhez nagyságrendekkel több adatra van szüksége az egyes algoritmusoknak

2.2.1. Nehézségi osztályok meghatározása - eddigiek

A Strava-ról gyűjtött utak sok adatot tartalmaznak azonban nincsenek kategorizálva, így cél az eddig megtett utakat nehézség alapján több osztályba sorolni, azonban ennek megvalósítása kérdéses. Mivel a helyes label-ek hiányoznak klaszterezési módszereket kell használni. Várhatóan az egyén adatai alapján és a összes összegyűjtött adat alapján készült osztályok különbözőek lesznek, hiszen más útvonalakat tettek meg, eltérő eredményekkel.

Egy általános modell elkészítése az elsődleges feladat, mivel az adatok sokszínűsége és mennyisége miatt jobb kiindulópontot jelent. Amennyiben egy adott felhasználónak megfelelően sok és sokféle útadata van akkor érdemes lehet az egyéni adatokra is kipróbálni az osztályozást.

2.2.2. Nehézség meghatározása - új útvonalra

A már megtett útvonalak osztályozása után a következő lépés az új, megtenni kívánt útvonalak nehézségének meghatározása. Ez megvalósítható egyrészt a már betanított klaszterezési algoritmusok felhasználásával, másrészt a klaszterezés által meghatározott osztályok label-jeit felhasználva valamilyen osztályozással.

2.2.3. Időtartam becslés

Az egyik legalapvetőbb feladat a várható idő meghatározása, hiszen ez az egyik legmeghatározóbb pontja a kerékpározásnak - mennyi időbe fog telni mire megtesz valaki adott kilométert? Ennek a kérdésnek a megválaszolására alapvetően a különböző regressziók alkalmasak, amik az út paraméterei alapján folytonos értéket adnak vissza. Azonban ez a pont kicsit bonyolultabb mint amilyennek tűnik, mivel alapvetően kétféle időtartamot különböztetünk meg: a mozgási és a tényleges időtartamot. Az első jelöli az összesen mozgással töltött időt, míg a második magába foglalja a különböző megállókat mint a pihenés, várakozás forgalmi lámpáknál, hosszabb utak esetén étkezéssel töltött időket stb. Ennek megoldására több lehetőség is kínálkozik.

A legegyszerűbb megoldás olyan regresszió használata amely egy inputhoz több eredmény, output-ot is tud társítani. Erre kínál lehetőséget a 2.3.1. pontban részletezett MLPRegressor. Ennek használatával rögtön mindkét idő előáll, nem szükséges további lépéseket tenni.

A második lehetőség több részből áll. Első lépésben tetszőleges regresszióval csak a mozgási időre készül becslés, mivel ez szorosabban összefügg az olyan alapvető adatokkal mint a távolság. Második lépésként pedig egy újabb regressziós modellt kell használni, ami a mozgási idő és egyéb paraméterek segítségével képes a teljes várható időre becslést készíteni.

2.2.4. Edzésterv

A felhasználók számára javasolt edzésterveknek több lehetséges megvalósítási módja is van. A legteljesebb kivitelezés a térképen jelölt konkrét útvonal, célul kitűzött teljesítési és részidőkkel, nehézségi szinttel. Azonban a szakdolgozat során egy valamivel egyszerűbb feladat elérése a cél, amit két alrészre lehet bontani.

- Az első típus esetén a felhasználó ad meg egy útvonalat, aminek adataiból kell a teljes várható időre, részidőkre, sebességre célkitűzéseket adni, alapul véve hogy a felhasználó milyen nehézségi szintet tűz ki. A részidőket érdemes külön vizsgálni, hiszen egy meredeken emelkedő szakasz teljesítése sokkal nagyobb kihívás a kerékpározók számára mint egy viszonylag sík terepen való haladás.

- A második típus esetén az út adatait is az algoritmusnak kell kiszámolnia, ideértve az út hosszát és a szintemelkedéseket. Ezen adatok valamint a kitűzött teljesítési idő egyensúlyba hozásával kell egy-egy nehézségi szinthez racionális nehézségű útvonal tervet létrehozni.

2.3. Gépi tanulás

Rengeteg gépi tanulási módszer létezik amik jól használhatóak különböző modellek, becslések, osztályok készítésére. Ezek a módszerek ma már legtöbbször egy programozási nyelv adott könyvtárának, csomagjának segítségével egyszerűen használhatóak. Ennek megfelelően a Python nyelv scikit-learn [6] csomagja adta a gépi tanulási módszerek alapját a szakdolgozat elkészítése során.

Az egyes algoritmusok használatán túl fontos a megfelelő adatforrás megtalálása, valamint az adatkinyerési és feldolgozási folyamat leegyszerűsítése, átláthatóvá tétele.

Ebben a fejezetben találhatóak meg a szakdolgozat során használt gépi tanulási módszertanok részletezése, valamint az adatgyűjtéshez létrehozott rendszer eszközeinek kifejtése.

2.3.1. Regresszió

A regresszió egy felügyelt gépi tanulási módszer amely folytonos kimenetet generál. A tanuláshoz szükség van a bemeneti adathalmazra valamint a helyes kimeneti értékekre, amik elemzésével az algoritmus képes új bemeneti adatpárokra kimenetet generálni.

Ridge Regression

A Ridge regresszió sajátossága hogy a túlilleszkedés kiküszöbölésének érdekében módosítja a költség funkció számítását. Ehhez egy "büntető" tényezőt, egy plusz súlyt használ, ezzel ellensúlyozva a hibákat.

$$\min_w \|Xw - y\|_2^2 + \alpha\|w\|_2^2$$

Itt $\alpha \geq 0$ egy komplexitási paraméter ami a csökkenés mértékét befolyásolja: minél nagyobb az α értéke annál gyorsabb a csökkenés.

Az algoritmus bonyolultsága egy $n \times p$ méretű X mátrix esetén $O(np^2)$ amennyiben $n \geq p$.

Lasso

A *Least Absolute Shrinkage and Selection Operator* angol kifejezés rövidítése.

$$\min_w \frac{1}{2n_{samples}}\|Xw - y\|_2^2 + \alpha\rho\|w\|_1$$

Elastic Net

TODO szöveges kifejtés

$$\min_w \frac{1}{2n_{samples}}\|Xw - y\|_2^2 + \alpha\rho\|w\|_1 + \frac{\alpha(1-\rho)}{2}\|w\|_2^2$$

MLPRegressor

Multi-layer Perceptron Regression, azaz mesterséges neuronokból felépülő többrétegű struktúrán, más néven egy neurális hálón alapuló regressziót valósít meg. Ez egy felügyelt tanulási algoritmus, tehát inputként a feature-ok egy mátrixát (X) valamint a

hozzájuk tartozó helyes label-eket (y) várja. Ezen értékek alapján backpropagation-t használva a módszer folytonos értékek készletével tér vissza. Az MLPRegressor továbbá több kimenetű regressziót is támogat, így egy mintához több output érték is rendelhető.

2.3.2. Osztályozás

Az osztályozás egy felügyelt gépi tanulási algoritmus, melynek lényege hogy adott input adatokból és helyes label-jeikből (amik megadják hogy az egyes adatok melyik osztályhoz tartoznak) tanulva az új adatokhoz megfelelő osztályt, csoportot rendeljen.

Nearest Neighbors Classification

A szomszéd-alapú osztályozás példányalapú tanulás: nem egy általános belső modellt próbál konstruálni, csak egyszerűen a training data példányait tárolja.

A scikit-learn csomag két különböző legközelebbi szomszéd módszert is kínál: a KNeighborsClassifier-t és a RadiusNeighborsClassifier-t

- **KNeighborsClassifier:** ez a módszer az egyes pontok k legközelebbi szomszédja alapján osztályoz, ahol k egy a fejlesztő által megadott egész szám. A k optimális értéke nagyban függ az adattól: általában egy nagyobb k elnyomja az anomáliákat, zajok hatását, azonban az osztály határok kevésbé lesznek egyértelműek. Ez a módszer elterjedtebb.
- **RadiusNeighborsClassifier:** az előző módszerrel ellentétben nem a k legközelebbi szomszédot veszi figyelembe, hanem a fejlesztő által megadott r sugárba eső pontokat csoportosítja egy osztályba. Amennyiben az adathalmaz nem egyenletesen mintavételezett a RadiusNeighborsClassifier jobb választás lehet.

Nearest Centroid Classifier

Egyszerű algoritmus, ahol minden osztályt a tagjai középpontja reprezentál

MLPClassifier

Multi-layer Perceptron Classification, azaz mesterséges neuronokból felépülő többrétegű struktúrán, más néven egy neurális hálón alapuló osztályozást valósít meg. Bemenetként a feature-ok X mátrixát és a label-ek y vektorát várja. A modell backpropagationt használ a tanulás során, pontosabban gradiens leereszkedést, ami backpropagation használatával kerül kiszámításra. Az osztályozáshoz a kereszt entrópia funkciót minimalizálja, az egyes x mintákhoz a $P(y|x)$ valószínűséget becsülve. Támogatja a multiclass osztályozást a Softmax függvény output funkcióként való alkalmazásával, valamint a multi-label osztályozást, melynek segítségével egy minta több osztályba is sorolható.

2.3.3. Klaszterezés

A klaszterezés alapfeladata hasonló az osztályozáséhoz, az adatokat különböző csoportokba kell rendezni, azonban ez felügyelet nélküli módszer, a helyes label-ek nélkül próbál adott számú osztályt kialakítani.

KMeans

TODO kifejtés

MeanShift

TODO kifejtés

2.4. A gépi tanulás alapköve - adatok

A gépi tanulás módszerei önmagukban mit sem érnek, megfelelő méretű és minőségű adathalmazra van szükség amiből az algoritmusok tanulni tudnak. Azonban adatgyűjtés mindig érzékeny terület, sok időt, energiát felemészt és a kívánt eredmény ezek befektetésével sem mindig garantált. A szakdolgozathoz elsősorban biciklivel megtett utak statisztikai adataira van szükség, amihez a Strava mobil és webes alkalmazás ad kiindulási alapot. A fejlesztői munkát segíti a Strava saját API-ja, aminek segítségével egyszerűen megvalósítható a felhasználók autentikációja valamint a szükséges adatok elérése.

2.4.1. Adatgyűjtés

A Strava által gyűjtött és tárolt adatok elérésének alapját egy, a szakdolgozat keretein belül készített weboldal adja. Ezen oldal felkeresésével a felhasználók tájékozódhatnak a szakdolgozatról, az adatgyűjtés fontosságáról valamint alapvető funkciókat használhatnak. Az oldalon lehetőség van új felhasználó létrehozására, már létező fiókokba való bejelentkezésre. Bejelentkezés után a felhasználó hozzákapcsolhatja a Strava fiókját a weblapon létrehozott fiókjához, ezzel lehetővé téve az útvonalai statisztikai adatainak olvasását. A weboldalon továbbá lehetőséget fog kínálni az elkészült, betanított program letöltésére, amivel a felhasználók is megtapasztalhatják az egyes algoritmusok hatékonyságát. A weboldal az alábbi eszközök felhasználásával valósult meg:

JQuery

A JQuery egy népszerű JavaScript könyvtár amely lehetőséget ad a HTML elemek könnyebb manipulálására, az események egyszerűbb kezelésére, animációk megvalósítására.

Firebase

A Firebase egy Google által fejlesztett sokoldalú rendszer, amely egyszerű eszközöket kínál egy weboldal logikai részének felépítéséhez. Nem alkalmas robosztus rendszerek létrehozásához, azonban kisebb projektek menedzseléséhez megfelelő alapot kínál. A projekteket létrehozás után online lehet kezelni és a Firebase funkcióit weboldalakban és mobil alkalmazásokban is (mind Android és IOs) egyszerűen lehet használni. A rendszer alapját egy Node.js szerver és egy JSON alapú adatbázis adja valamint elérhető egy általános célú tárhely tetszőleges file-ok tárolásához.

Firebase - Autentikáció

A szakdolgozathoz készített weboldal szempontjából a Firebase egyik legfontosabb eleme az autentikációs rendszer. A Firebase projektben való engedélyezés után néhány sor JavaScript kóddal gyorsan és biztonságosan megvalósítható az új felhasználók regisztrálása, bejelentkeztetése. Az email alapú regisztráláson túl lehetőség van külső rendszerekkel való autentikációra is mint a Google fiók, Facebook, Github. Továbbá elérhető a már regisztrált felhasználók emailen kersztül történő értesítése is.

Firebase - Adatbázis

A Firebase alapvetően ingyenes, emiatt az adatbázisnak vannak méret és sebességbeli korlátjai, azonban megfelelő kiindulási alapot ad. A JSON alapú struktúra megkönnyíti az adatok kliensen való kezelését valamint egy ilyen méretű alkalmazás esetén egyszerűen karban tartható. Fontos kiemelni az adatbázis hozzáférhetőségének konfigurálási lehetőségét. Lehetőség van az adatbázis egészére vagy a részfákra olvasási és írási beállításokat megadni, amik korlátozhatóak a bejelentkezett felhasználó azonosítója alapján is. Ennek felhasználásával könnyen megvalósítható hogy a rendszer összes felhasználója csak a saját adataihoz férjen hozzá ami fontos adatvédelmi szempont.

2.4.2. Adatstruktúra

A szakdolgozathoz készített weboldal fő célja hogy lekérje és tárolja azon felhasználók útvonalait akik regisztrálás után kapcsolódtak a Strava fiókjukhoz. A Strava API-ja az utak lekérésekor SummaryActivity objektumok tömbjével tér vissza, ahol minden objektum 39 változóval jellemez egy utat, aktivitást. Ez a 39 jellemző magába foglalja többek között a felhasználó azonosítóját, a megtett távot, a teljesített kihívások számát, az ismerősök megjegyzéseinek a számát és rengeteg egyéb információt. A gépi tanulás során ezeknek a jellemzőknek csak egy részét érdemes használni, amik szorosabban kapcsolódnak az út valós leírásához. Ezek a jellemzők a következők:

- **average_speed:** átlagsebesség (méter/másodperc)
- **average_watts:** átlagos erőkifejtés az útvonal során. Amennyiben nem biztosított az értéke -1
- **commute:** egy tevékenység rögzítésekor lehetőség van azt 'ingázásként' felcímkézni, ezzel megjelölve hogy nem egy verseny vagy túra adat, hanem mindennapi, például munkába vagy boltba járási útvonal. Feltételezhető hogy ilyenkor az illető nem a minél jobb eredmények elérésére törekszik, inkább rutin szerűen, átlagos tempóban halad (Logikai érték, -1 amennyiben nem biztosított)
- **device_watts:** azt jelöli hogy az erőkifejtési adatokat tényleges mérőkészülék szolgáltatja vagy csupán becsült érték (logikai, Hamis amennyiben becsült érték)
- **distance:** a megtett távolság (méter)
- **elapsed_time:** összesen eltelt idő (másodperc)
- **elev_high:** az út legmagasabb pontja (méter)
- **elev_low:** az út legalacsonyabb pontja (méter)

- **end_latlng:** az útvonal végpontjának hosszúsági és szélességi helyzete
- **flagged:** az útvonal meg van e jelölve (logikai)
- **has_heartrate:** elérhető e pulzus adat (logikai)
- **heartrate_opt_out:**
- **kilojoules:** összes kifejtett erő mennyisége (kilojoule)
- **max_speed:** maximum sebesség az út során (méter/másodperc)
- **max_watts:** maximális erőkifejtés az út során
- **moving_time:** mozgással töltött idő (másodperc)
- **pr_count:**
- **resource_state**
- **start_date_local:** az út kezdési ideje (dátum)
- **start_latlng:** az útvonal kezdőpontjának hosszúsági és szélességi helyzete
- **total_elevation_gain:** összesen megtett emelkedő (méter)
- **trainer:** az útvonal gépen lett e megtéve
- **weighted_average_watts:**
- **workout_type:** tevékenység típusa, lehet edzés vagy verseny (egész szám, ha nincs megadva akkor -1 ??)

3. fejezet

Fejlesztői dokumentáció

Az Elméleti kifejtés során meghatározott célokat különböző gépi tanulási módszerekkel lehet megoldani amik eltérő eredménnyel, pontossággal fognak működni. A nehézségi osztályok meghatározására elsősorban a a 2.3.3. pontban részletezett klaszterezési módszerek használhatóak

3.1. Adathalmaz előkészítés

A fejlesztés legelső lépése az összegyűjtött adatok megfelelő struktúrára való átalakítása és megtisztítása, ezzel megkönnyítve a későbbi feldolgozást valamint növelve a gépi tanulási algoritmusok pontosságát

3.1.1. Adatstruktúra kialakítása

Az adatgyűjtés elsődleges eszközeként a szakdolgozat keretein belül készített weboldal szolgált, amely minden információt egy JSON alapú adatbázisban tárolt le. Az adathalmaz előkészítésének a legelső lépése a JSON struktúráról való áttérés egy, a Python programozási nyelv által kezelt, könnyen használható adatstruktúrára. Erre a szakdolgozat során a Pandas [17] nevű Python csomag által megvalósított DataFrame nevű struktúra fog szolgálni , amely segítségével az adatokat táblázathoz hasonló formában lehet tárolni. Egy DataFrame oszlopai egyedi, a fejlesztő által definiált oszlopnevekkel érhetőek el, soraira index használatával lehet hivatkozni. Az oszlopok egyenként különböző típusúak lehetnek és tartalmazhatnak NULL értékeket. A DataFrame egyik legnagyobb előnye az oszlopok nevesítése, amivel könnyen nyomon követhető hogy az aktuális értékek melyik feature-nek felelnek meg, mit reprezentálnak a valóságban

3.1. Program részlet. A parancs segítségével a JSON struktúra (jsonData) könnyen átalakítható DataFrame-é (dataset). A dataset oszlopai megfelelnek a a 2.4.2. pontban részletezett adatoknak.

```
1 import pandas
2
3 jsonData = pandas.read_json("2019_02_08_10h.json", type='series')
4 cleanData = []
5 for u in userData:
6   if userData[u]['connectedToStrava'] == True:
7     for i in range(0,len(userData[user]['activities'])):
```

```
8           userData[u]['activities'][i]['ageGroup'] = userData[u]['ageGroup']
9           userData[u]['activities'][i]['sex'] = userData[u]['sex']
10          userData[u]['activities'][i].pop('external_id', None)
11          userData[u]['activities'][i].pop('map_id', None)
12          userData[u]['activities'][i].pop('map_resource', None)
13          userData[u]['activities'][i].pop('map_summary', None)
14          cleanData.append(userData[u]['activities'][i])
15      else:
16        print('User does not have any activities')
17  dataset = pandas.DataFrame(cleanData)
```

A fenti parancs segítségével a JSON struktúra (jsonData) könnyen átalakítható DataFrame-é (dataset). A dataset oszlopai megfelelnek a a 2.4.2. pontban részletezett adatoknak.

3.1.2. Adattisztítás

A megfelelő adatstruktúra kialakítása után a következő lépés az adatok megtisztítása. Ez a lépés azért szükséges mert a legfigyelmesebben gyűjtött adathalmaz is tartalmazhat rossz értékeket illetve előfordulhat hogy több helyen hiányzik a valódi érték. Ezeknek a hibáknak a megtalálása és kijavítása több fázisból áll.

features	count	mean	std	min	0.25	0.50	0.75
age_group	10014	1.01	0.48	0.0	1.0	1.0	1.0
average_speed	10014	24.25	653.31	0.0	4.96	5.73	6.47
average_watts	1848	163.0	49.51	0.0	139.1	161.85	180.85
distance	10014	26153.09	29465.23	0.0	7444.83	13751.25	34587.2
elapsed_time	10014	5845.88	19296.2	0.0	1648.0	3020.0	7343.0
elev_high	9845	282.1	272.66	-71.2	162.2	229.4	296.4
elev_low	9844	131.99	68.68	-500.0	103.1	130.0	148.3
kilojoules	1765	1295.76	756.06	0.0	728.7	1236.7	1745.4
max_speed	10014	12.14	3.99	0.0	9.6	11.6	14.48
max_watts	546	630.95	244.34	115.0	502.0	603.0	708.0
moving_time	10014	4543.18	5016.52	0.0	1442.0	2451.0	6109.0
pr_count	1959	1.48	3.2	0.0	0.0	0.0	2.0
resource_state	1959	2.0	0.0	2.0	2.0	2.0	2.0
total_elevation_gain	10014	278.7	428.88	0.0	27.5	85.2	361.8
weighted_average_watts	546	188.86	30.35	5.0	176.25	195.0	208.75
workout_type	4039	9.74	1.85	0.0	10.0	10.0	10.0

NULL értékek

Egy általános adatbázis esetén gyakran előfordul hogy egyes oszlopok NULL értékeket tartalmaznak - ebben az esetben például NULL jelöli ha egy útvonal nincs elérhető

adat egy adott jellemzőről. Azonban a gépi tanulási algoritmusok számokat képesek feldolgozni így elengedhetetlen a NULL értékek kiküszöbölése valamilyen formában.

A NULL értékek kiküszöbölésére két elterjedt megoldás létezik:

- **NULL értékek törlése:** az egyszerűbb megoldás a NULL értékeket tartalmazó sorok törlése, azonban ennek a módszernek nagy hátránya hogy sok NULL-t tartalmazó adathalmaz esetén jelentős mértékben megcsappan az adathalmaz mérete
- **NULL értékek feltöltése:** összetettebb, azonban sok esetben célravezetőbb megoldást jelent a NULL értékek helyettesítése valamilyen számított értékkel. Az új értékek számítása különböző módokon történhet, ez nagyban függ az adott jellemző jellegétől

Amennyiben egy oszlop nagy mértékben tartalmaz NULL értékeket érdemes megfontolni az elvetését vagy külön esetként kezelni mikor van tényleges érték

Az adat vizsgálata után az első szembetűnő gond a hiányzó értékek. Az alábbi jellemzők akkora mértékben hiányosak hogy a javításuk reménytelen feladat, így az adattisztítás elején törlésre kerültek.

- average_watts: 18.45% hasznos értéket tartalmaz
- commute: 98.83% hasznos értéket tartalmaz azonban ezekből csupán 23 True érték
- device_watts : 19.56%
- flagged : 19.56%
- has_heartrate : 19.56%
- heartrate_opt_out:
- kilojoules : 17.63%
- max_watts 5.54%
- pr_count : 19.56%
- resource_state : 19.56%
- weighted_average_watts : 5.45%
- sex : 100% hasznos azonban elenyészően kevés női adatot tartalmaz az adathalmaz, ezért a torzítatlanság érdekében a szakdolgozat során csak a férfi útvonalak kerülnek felhasználásra - így ez az jellemző funkcióját veszti, konstans érték
- start_latlng : 18.34%
- end_latlng : 18.34%

Kiugró értékek

Gyakori jelenség hogy egy jellemző tartalmaz néhány magasan kiugró értéket, amelyek sokszor érvényesek azonban fakadhatnak mérési / rögzítési hibából is. Akár érvényes értékek, akár valamilyen hibából erednek érdemes kiküszöbölni őket mivel könnyen eltorzíthatják az eredményeket.

A kiugró értékek megtalálása egy alapvető módszer került felhasználásra a szakdolgozat készítése során. A kiugró értékeket jellemzőként külön vizsgáljuk. F jellemzőre $x \in F$ érték kiugrónak tekinthető amennyiben

$$\frac{x - \overline{F}}{\sigma} > 2.5$$

teljesül, ahol $\overline{F}$ az F jellemző átlaga, σ pedig F szórása.

Az alábbiakban a különböző jellemzők kiugró érték vizsgálatának jellemzése található.

average_speed (átlagsebesség):

Ez a jellemző m/s-al adja meg az útvonal átlagsebességét. A fenti módszer 10 kiugró értéket talált amik 2182 m/s (7855.2 km/h) és 28290 ms/s (101844.0 km/h) közé esnek. Ekkora sebesség elérése nyilvánvalóan lehetetlen kerékpárral, eredetük az adatsorok megvizsgálása után leszűkíthető a mozgási idő (moving_time) jellemző mérésének a hiányára / hibájára. Az átlagsebesség számítása ugyanis a távolság és a mozgási idő hányadosaként kerül kiszámításra és a vizsgált kiugró értékeknél a mozgási idő mindenhol 1 másodperc míg a távolság 10 km fölött van. Ezen adatok javítása nem tűnik megvalósíthatónak. Eldobásuk után az adathalmaz 10004 utat tartalmaz.

distance (távolság):

A distance jellemző méterben megadva tárolja az egy-egy útvonal során megtett távolságot. Kiugró érték vizsgálat során 313 értéket talált a módszer, amik között a legkisebb értéke 99915.1 méter azaz közel 100 km. Ezek az adatsorok azonban minden más szempontból helyes értékeket tartalmaznak és valójában a 100 km-s utak sem lehetetlenek. Ezek az értékek nem kerültek törlésre az adathalmazból azonban a későbbiek során érdemes lehet a hasonlóan hosszú utakat külön kezelni

elev_high (legmagasabb tengerszint feletti pont):

Az elev_high jellemző vizsgálata során 61 kiugró érték jelzett a módszer amelyek 976.6 méter és 12103.0 méter közé esnek. A 12103 méter egyértelműen lehetetlen, a többi érték pedig vele együtt elvetésre kerül. Ezen adatsorok eldobása után az adathalmaz 9943 útvonalat tartalmaz

max_speed (legnagyobb sebesség)

A max_speed az útvonal során mért maximális sebességet jelzi m/s mértékegységben. Vizsgálata során a használt módszer 218 kiugró értéket talált. Ezek az értékek két intervallumra oszthatóak fel. 0.0 m/s - 2.1 m/s és 22.1 m/s - 122.7 ms azaz 0.0 km/h - 7.56 km/h és 79.56 km/h - 441.72 km/s. Ezek olyan adatok amelyek kilógnak az adathalmazból, sok közülük lehetetlen / értelmezhetetlen. Ezen adatsorok eldobása után az adathalmaz 9725 útvonalat tartalmaz.

moving_time (mozgási idő)

A moving_time jellemző a tényleges mozgással töltött időt tárolja másodpercben megadva. A jellemző vizsgálata során 276 érték bizonyult kiugrónak amelyek közül a legkisebb értéke 16814 másodperc azaz nagyjából 4 óra és 40 perc. Ezek az értékek összefüggnek a távolság (distance) jellemző vizsgálata során talált kiugró értékekkel, így egyenlőre az adathalmaz része marad, a későbbiek során azonban külön lesz kezelve.

A kiugró értékek megtalálására használt módszeren felül definiálásra került néhány szabály amik segítségével ki lehet szűrni az egyéb helytelen adatokat. Ezek alapján elvetésre kerülnek azok az adatsorok ahol:

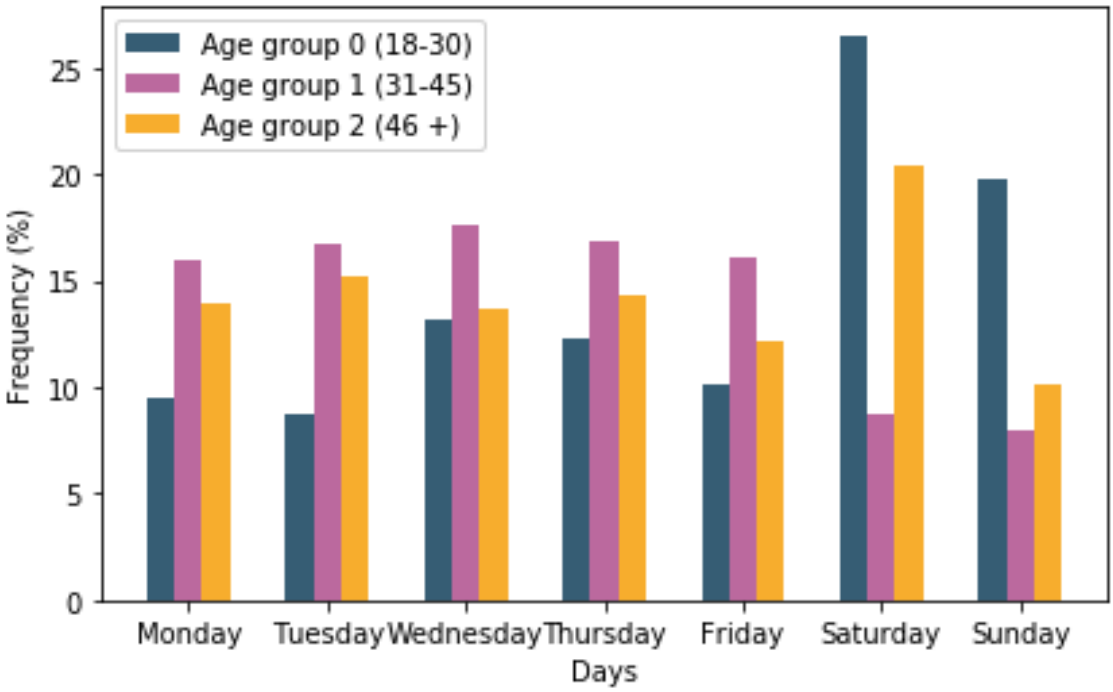
- az átlagsebesség kisebb mint 2 m/s (7.2 km/h)
- a távolság kisebb mint 100 méter (néhány száz méter esetén előfordulhat hogy rövid sprinteket tettek meg a sportolók)
- a legmagasabb vagy legalacsonyabb tengerszint feletti magasság hiányzó adat

Továbbá 2 helyen javításra került az összes eltelt idő (elapsed_time) mivel kevesebb volt mint a mozgással tölött idő (moving_time)

A fentebbi adatsorok eldobása után összességében 9603 útvonalat tartalmazó adathalmaz maradt.

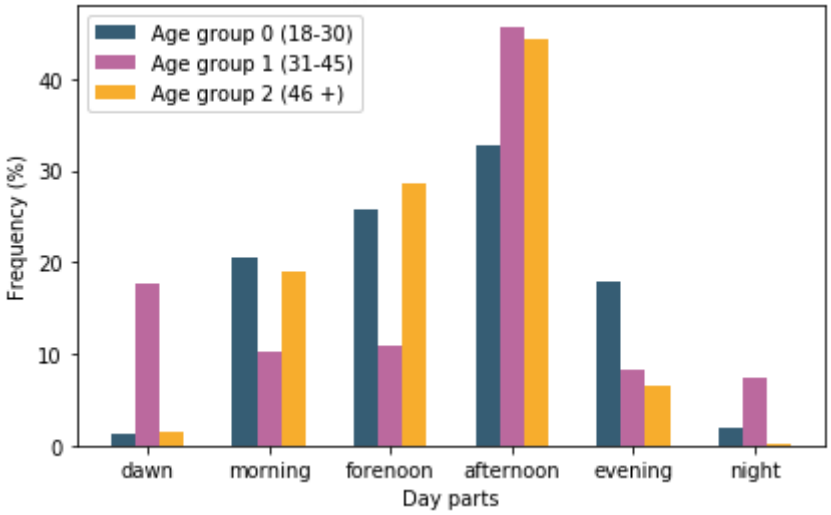
3.1.3. Adatok áttekintése

A további feldolgozás illetve a tényleges gépi tanulási algoritmusok felhasználása előtt érdemes kicsit mélyrehatóbban tanulmányozni az adathalmazt, felfedezni a mintákat és összefüggéseket.

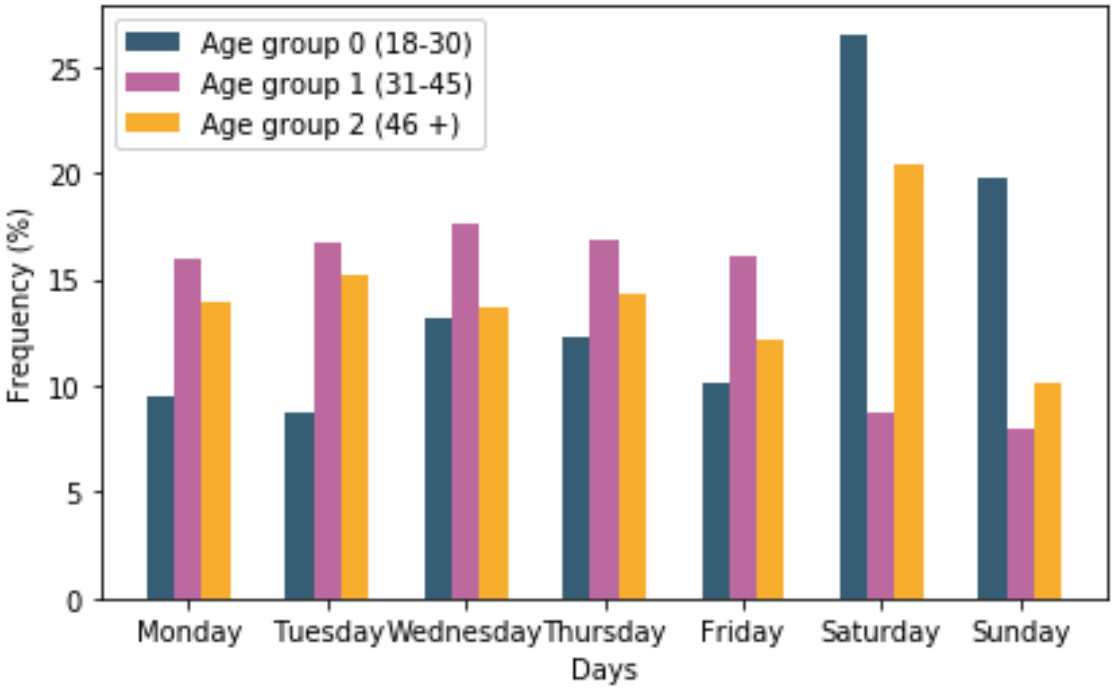


3.1. ábra. Activity frequency by days of week.

Szöveg a fenti képről....
Blablabla
Még mindig.....
Szöveg a fenti képről....
Blablabla
Még mindig.....
Szöveg a fenti képről....

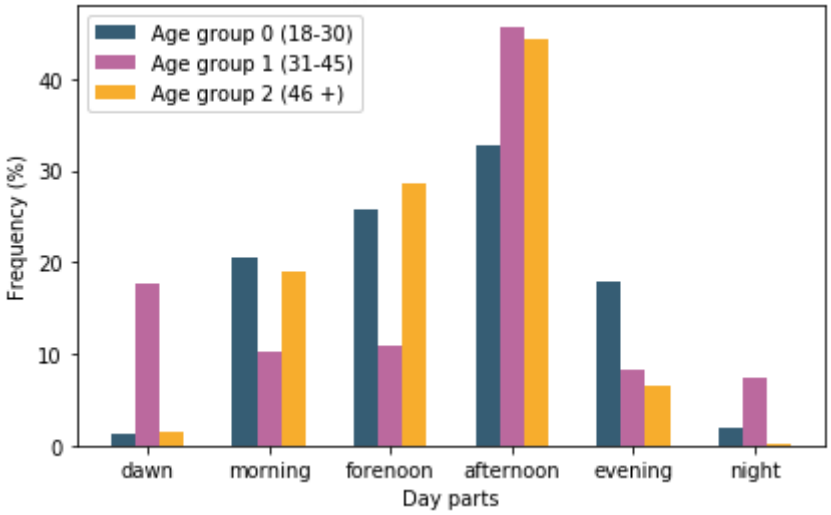


3.2. ábra. Activity frequency by dayparts.



3.3. ábra. Activity frequency by days of week.

Blablabla
Még mindig.....
Szöveg a fenti képről....
Blablabla
Még mindig.....



3.4. ábra. Activity frequency by dayparts.

3.1.4. További adat feldolgozás

One-hot encoding

A kiugró értékek kezelése és a adathalmaz összefüggéseinek vizsgálata után a következő lépés a one-hot encoding. Az eljárás lényege hogy egy kategória alapú jellemzőt több (a kategóriák számának megfelelő) oszlopra bont szét, ahol az egyes oszlopokban, jellemzőkben 1-es szerepel amennyiben az adatsor abba az adott kategóriába sorolható, 0 egyébként. Ennek az előnye hogy kiküszöböli a kategóriák sorszámmal történő jelzésének távolságát. Például egy korcsoportot tároló jellemző esetén nem biztos hogy helyes feltételezni hogy a 0. és a 2. kategória (csoport) között 2 távolság van. A one hot encoding természetesen folytonos változók esetén nem értelmezhető és kategoriális jellemzőkre sem mindig érdemes alkalmazni.

A szakdolgozathoz használt adathalmaz esetén 3+1 jellemző kerül one-hot encodeolásra: age_group (korcsoport), workout_tpye, start_date_local és a training.

age_group: a sportoló korcsoportját jelzi, értékei egy három elemű halmazból kerülhetnek ki: {0,1,2} ahol 0 a [18,30], 1 a [31,45] 2 pedig a [46,+∞] korcsoportokba való tartozást jelenti. One-hot encoding során az eredeti jellemzőből 3 oszlop készül, végül az eredeti törlésre kerül az adathalmazból.

3.2. Program részlet. Python parancsok részletezése

```
1 ageOneHot = pandas.get_dummies(rawData['age_group'], prefix='age')
2 rawData.drop(columns=['age_group'], inplace=True)
3 rawData = rawData.join(ageOneHot)
```

A kódban megadott prefix paraméter segítségével az új jellemzők nevei: age_0.0, age_1.0, age_2.0

workout_tpye:

3.3. Program részlet.

```
1 workoutTypeOneHot = pandas.get_dummies(rawData['workout_type'], prefix='workout_type')
2 rawData.drop(columns='workout_type', inplace=True)
3 rawData = rawData.join(workoutTypeOneHot)
```

start_date_local: a jellemző az aktivitás kezdeti idejét tárolja, a sportoló saját időzónája szerint, ÉÉÉÉ-MM-DD HH:MM:SS formátumban. Dátum mezők azonban nem értelmezhetőek gépi tanulási algoritmusokkal így a jellemző átalakítása szükségszerű volt. A dátumból alapvetően 2 féle új jellemző kerül kinyerésre, majd ezek külön kerülnek one-hot encode-olásra

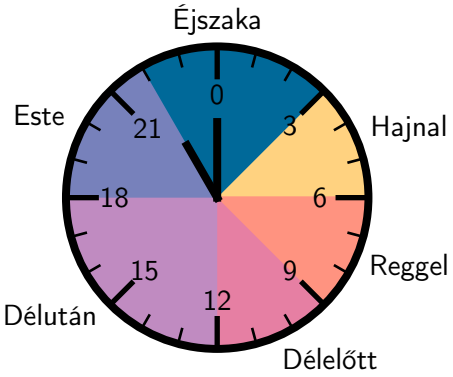
3.4. Program részlet. Az aktivitás kezdetének dátumából kinyerhető hogy a hét melyik napján történt az aktivitás. Ennek az információnak a kinyerését és one-hot encode-olását az alábbi kód végzi

```
1 rawData['day_of_week'] = rawData['start_date_local'].dt.day_name()
2 weekDayOneHot = pandas.get_dummies(rawData['day_of_week'],
3                                    prefix='weekday')
4 rawData.drop(columns=['day_of_week'], inplace=True)
5 rawData = rawData.join(weekDayOneHot)
```

3.5. Program részlet. Az aktivitás kezdetének időpontjából kinyerhető hogy melyik napszakban kezdődött az aktivitás. Ennek az információnak a kinyerését és one-hot encode-olását az alábbi kód végzi

```
1 rawData['daypart'] = rawData['start_date_local'].apply(lambda row:
2                                                        timeToPartOfDay(row))
3 dayPartOneHot = pandas.get_dummies(rawData['daypart'], prefix='daypart')
4 rawData.drop(columns=['daypart'], inplace=True)
5 rawData = rawData.join(dayPartOneHot)
```

Ahol a timeToPartOfDay függvény a start_date_local jellemző minden adattagjára külön hívódik meg. Ez a függvény szubjektíven oszt fel egy napot hat napszakra a 3.5. ábra szerint



3.5. ábra. Napszakok kialakítása.

trainer: a trainer jellemző Igaz / Hamis értékekkel jelzi hogy egy adott aktivitás traineren, azaz gépen történt e, nem pedig tényleges kerékpáron. Ennek a jellemzőnek az átalakítása nem igazi one-hot encoding, inkább csak az Igaz / Hamis értékek átmappelése 1 / 0 értékekre.

Az adattisztítás összes lépése után az adathalmaz szerkezete lényegesen megváltozik, új jellemzőket tartalmaz (one-hot encoding) valamint az eddigi jellemzők sok esetben más intervallumokba esnek (kiugró értékek kezelése).

Jellemzők:

- average_speed
- distance
- elapsed_time
- elev_high
- elev_low
- hashed_id
- max_speed
- moving_time
- total_elevation_gain
- age_0.0
- age_1.0
- age_2.0
- trainer_onehot
- workout_type_0.0
- workout_type_4.0
- workout_type_10.0
- workout_type_11.0
- workout_type_12.0
- weekday_Friday
- weekday_Monday
- weekday_Saturday
- weekday_Sunday
- weekday_Thursday
- weekday_Tuesday',
- weekday_Wednesday',
- daypart_afternoon',
- daypart_dawn',
- daypart_evening',
- daypart_forenoon',

- daypart_morning',
- daypart_night

az adathalmaz mentésre kerül a későbbi felhasználás megkönnyítésének céljából. A tisztított adatot a *data_advanced_date.csv* fájl tartalmazza

3.2. Gépi tanulás

3.2.1. Mozgási és eltelt idő számítása

Az adathalmaz egy aktivitásra tekintve két különböző időtartamot különböztet meg: a mozgással töltött (moving_time) és az összesen eltelt (elapsed_time) időt, mindkettőt másodpercben mérve. A szakdolgozat során ennek a két időtartamnak a várható értékének a meghatározása az egyik cél gépi tanulási algoritmusokkal valamint ezen algoritmusoknak az összehasonlítása. Ennek a feladatnak a megoldásához a 2.3.1. alfejezetben kifejtett regressziók adnak alapot.

A kétféle időtartam alapvetően külön kezelve, egyenként kerül becslésre, erre kivételt fog jelenteni a **TODO** href pontban részletezett MLPRegressor.

Mozgási idő (moving_time)

A tisztított adathalmaz beolvasása után szükséges kialakítani az X és y halmazokat amelyek a **TODO** adatokat tartalmazzák. Az y fogja tartalmazni a moving_time jellemzőt míg X minden mást ami alapján y megbecsülhető.

A regressziók során nagyon fontos az adathalmaz skálázása, egy intervallumra hozása. Ennek fontossága akkor mutatkozik meg igazán mikor a különböző jellemzők nagyon eltérő intervallumokba esnek, például az average_speed 2.03 és 12.478 értékek között mozog míg a distance 161 és 436806 között. Ezért a gépi tanulás során gyakran alkalmazott eljárás az adathalmaz normalizálása vagy standardizálása amely az összes jellemzőt egy szűk intervallumra vetíti le. A standardizálás elvégzésére a szakdolgozat során a scikit-learn csomag által biztosított StandardScaler függvény szolgált

3.6. Program részlet. szöveg

```
1 from sklearn.preprocessing import StandardScaler
2 X = dataset.copy()
3
4 names = X.columns
5 scaler = StandardScaler()
6
7 scaledData = scaler.fit_transform(X)
8 scaledData = pandas.DataFrame(scaledData, columns=names)
```

3.7. Program részlet. A túltanulás elkerülése érdekében a tanító és cél adathalmazokat szét kell osztani tanító és tesztelő részekre. A regressziós modellek csak a tanító (train) adatokon fognak tanulni, ellenőrzésre, pontosság meghatározására pedig a teszt (test) adatok szolgálnak. **TODO**

```
from sklearn.model_selection import train_test_split

scaledY = scaledData['moving_time']
scaledData.drop(columns=['moving_time','elapsed_time'], inplace=True)

trainX, testX, trainy, testy = train_test_split(scaledData, scaledY,
                                                random_state=42)
```

Ridge regresszió: ez az algoritmus 2 paraméterrel finomhangolható: α és solver. **TODO** kifejtés. A moving_time esetén a standardizált adathalmazra a legoptimálisabb (legnagyobb pontosságot eredményező paraméterek) a $\alpha = 0.003$ solver : saga. Ezeknek az alkalmazása esetén, a modellt a trainX és trainy adathalmazokon tanítva a pontosság 96.251% (a teszt adatokon). A modell nem tanul túl, mivel a teszt adatokon (amiket a modell tanulás során nem ismer) nagy pontosság jellemzi a regressziót.

Lasso regresszió: $\alpha = 0.0001$, pontosság: 96.246%

4. fejezet

Összefoglalás

Ebben a fejezetben kell összefoglalni a szakdolgozat eredményeit, sajátosságait és a témában való elhelyezkedését. A fejezet címe nem módosítható! Lehet benne több alfejezet is, de nem ajánlott. Min 1 max 4 oldal terjedelemben

Irodalomjegyzék

[1] http://bikecalculator.com/

[2] Strava sporttevékenység követő alkalmazás. Strava.com

[3] Endomondo

[4] https://www.bestbikesplit.com/

[5] Firebase

[6] @articlescikit-learn, title=Scikit-learn: Machine Learning in Python, author=Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., journal=Journal of Machine Learning Research, volume=12, pages=2825–2830, year=2011

[7] Scikit-learn Ridge regression

[8] Scikit-learn Lasso

[9] Scikit-learn Elastic net

[10] Scikit-learn MLPRegressor

[11] Scikit-learn KNeighbors Classifier

[12] Scikit-learn Radius Neighbors Classifier

[13] Scikit-learn Nearest Centroid Classifier

[14] Scikit-learn MLPClassifier

[15] Scikit-learn KMeans

[16] Scikit-learn MeanShift

[17] Pandas: Python csomag

[18] cikk: https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57

Adathordozó használati útmutató

Ebben a fejezetben kell megadni, hogy a szakdolgozathoz mellékelt adathordozót hogyan lehet elérni, milyen struktúrát követ. Minimum 1 max 4 oldal. Lehet több alszakasz. Fejezet címe nem módosítható