

SZAKDOLGOZAT



MISKOLCI EGYETEM

Gépi tanulási módszerek alkalmazása és
összehasonlítása sportteljesítmény adatok
elemzéséhez

Készítette:

Ferencsik Dorina Kinga
Évfolyam. szak-szak

Témavezető:

Dr. Baksáné Dr. Varga Erika
egyetemi docens

MISKOLC, 2019

MISKOLCI EGYETEM

Gépészszmérnöki és Informatikai Kar
Alkalmazott Matematika Tanszék

Szám:

SZAKDOLGOZAT FELADAT

Ferencsik Dorina Kinga (YXI8DJ) programtervező informatikus jelölt részére

A szakdolgozat tárgyköre: adatbányászat

A szakdolgozat címe: Gépi tanulási módszerek alkalmazása és összehasonlítása sport-teljesítmény adatok elemzéséhez

A feladat részletezése:

A szakdolgozat témája a gépi tanulási módszerek összehasonlítása és alkalmazási lehetőségeinek kipróbálása sport teljesítmény adatokon. Az adatok forrása a Strava adatgyűjtő alkalmazás, amely futás és kerékpározás során rögzíti a sportoló teljesítményét és a sportolás után alapvető statisztikai adatokat szolgáltat (megtett km, átlag sebesség, stb.). Az adatok felhasználásával nemenként és korcsoportonként külön modellt kell felállítani, amelyeknek meg kell határozni a paramétereit. Ezek ismeretében a modellek alkalmasak lesznek előrejelzés készítésére. A dolgozat keretén belül elkészül majd egy tesztrendszer, ami szemlélteti az egyes módszerek pontosságát.

Témavezető: Dr. Baksáné Dr. Varga Erika, egyetemi docens

A feladat kiadásának ideje: 2018. szeptember 20.

.....
szakfelelő

EREDETISÉGI NYILATKOZAT

Alulírott; Neptun-kód:
a Miskolci Egyetem Gépészmeérnöki 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 szak-
irodalom 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 véleges eredménye:

Miskolc,

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. Bevezetés	7
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	12
2.2.1. Nehézségi osztályok meghatározása	12
2.2.2. Időtartam becslés	12
2.2.3. Edzésterv	12
2.3. Gépi tanulás	13
2.3.1. Regresszió	13
2.3.2. Klaszterezés	14
2.4. A gépi tanulás alapköve - adatok	16
2.4.1. Adatgyűjtés	16
2.4.2. Adatstruktúra	18
3. Adatfeldolgozás és gépi tanulás	20
3.1. Adathalmaz előkészítés	20
3.1.1. Adattisztítás	20
3.1.2. Adatok áttekintése	23
3.1.3. További adat feldolgozás	25
3.2. Regresszió	28
3.2.1. Paraméter hangolás	28
3.2.2. Mozgási idő	30
3.2.3. Összesen eltelt idő	31
3.3. Klaszterezés	33
3.3.1. 0. korcsoport	34
3.3.2. 1. korcsoport	36
3.3.3. 2. korcsoport	37
3.3.4. Összefoglalás	38
4. Összefoglalás	41
Irodalomjegyzék	42

1. fejezet

Bevezetés

A gépi tanulás egy egyre szélesebb körben használt és alkalmazott informatikai ág, mely az egyszerű levélszemét szűréstől a képfelismerésen és hangfeldolgozáson át rengeteg területen fontos szerepet játszik. Célja mindenkor meglévő adatok elemzése, egyfajta tanulási folyamat, amely után a jövőben új adatokra valamelyen becslés készíthető. Ennek az ágazatnak az al-területei nagyon szétágazóak, rengeteg létező megvalósítással. Általánosságban a tanulási módszer és a cél szerint csoportosíthatóak:

- felügyelt tanulás: az adatnak része az információ amelynek a becslése a cél. A felügyelt módszerek csoportjába tartozó algoritmusok a magyarázó jellemzőket tartalmazó X adathalmaz alapján tanulják meg a függő y változó értékeit, általában valamelyen közelítést alkalmazva. A gépi tanulási módszerek nagyobb része ide sorolható.
- felügyelet nélküli tanulás: ebben az esetben az adathalmaz nem tartalmazza a függő y változót, nem tudni hogy mire kell becslést készíteni. Jellemzően ide tartoznak a különböző klaszterezési eljárások, ahol valamelyen szempontból hasonló adatpontok csoportokba rendezése a cél, anélkül hogy az adatok hovatartozásáról bármilyen információ ismert lenne.

A különböző gépi tanulási módszerek terjedésével a sport világában is hamar megjelent az alkalmazásuk, azonban elsősorban a profi, versenyzői szférában ahol régebb óta általános a sportolói eredményekre vonatkozó adatok gyűjtése, valamint természetesen nagyobb a pénzügyi forgás mint a hobbi sportolók esetében. Szakdolgozatom célja ebből kifolyólag a minden nap, hobbi sportolók eredményeinek feldolgozása gépi tanulási módszerekkel, hiszen ez egy olyan terület ahol még nem sok megoldás létezik, azonban egyre nagyobbak a felhasználói igények.

A konkrét megvalósításhoz kerékpáros teljesítmények felhasználása került kitűzésre, egyrészt saját érdeklődési kör miatt, másrészt a kerékpáros útvonalak egyszerű mérhetőségéből és az adatok sokszínűségből kifolyólag. Az így kitűzött adathalmaz specifikusságából eredően az adatgyűjtési folyamat is a szakdolgozat részét képezi, ami magába foglalja egy erre a céllra készített weboldal megtervezését és kialakítását.

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 sporttevékenységeket 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. Előrelépések hozzájárulnak 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 feszeggetve 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 számára.

2.1.3. Endomondo

Az Endomondo [3] webes és mobil alkalmazás sporttevékenység nyomon követését hivatott megkönnyíteni a felhasználói számára. Rengeteg különböző sportágot 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 lehetőségeük 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 eszközükkel kellene figyelniük.

2.1.4. Best Bike Split

A Best Bike Split [4] egy rendkívül összetett rendszer amely elsősorban versenyzők számára kínál széles funkcionálitást. 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 tulajdonsgait (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.

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, tárolásukkal könnyen nyomon követhető a személyes fejlődés, kitűzhetők heti, havi, éves célok. Az útvonal paramétereinek nyomon követése, struktúrája és tárolása megvalósításunként változó, azonban a cél minden 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. Ezekben 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, adott évre vonatkozó, nem és korcsoport alapján lebontott) ahol lehetősége van ismerőkkel és ismeretlenekkel vetélkedni ezzel felbresztve 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 kerékpározni. 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, pontosságának alapjául az eddig megtett útvonalak és az elért eredmények szolgálhatnak, 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 minden 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.

A fenti csoportok összefoglalása a vizsgált alkalmazásokra:

	Bike calculator	Strava	Endomondo	Best Bike Split
Útvonal rögzítés	✗	✓	✓	✓
Közösségi funkciók	✗	✓	✓	✗
Tervezés (általános)	✓	✓	✓	✓
Tervezés (egyéni)	✗	✓	✗	✓

Célkitűzés

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, így a szakdolgozat során összehasonlításra kerülő gépi tanulási algoritmusok célja sportolói teljesítmény becslése. A teljesítményt természetesen nagy mértékben befolyásolja a kerékpáros felszerelések minősége és állapota, az útvonal terep és időjárás viszonyai azonban az egyén általános és pillanatnyi fizikai és szellemi állapota (kimerültség, kedvtelenség stb.) nem elhanyagolható tényező. Az átlagos értelemben vett teljesítmény több részre bontható le a kerékpározás kapcsán, melyek a szakdolgozat során külön kerülnek megválasztásra.

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 (álmoság, kimerültség, kedvtelenség stb.). Ezért el kell döntenи 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

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 különbözők 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 felhasználó megfelelően sok és sokféle útvonal adattal rendelkezik akkor érdemes lehet egyéni, személyre szabott nehézségi csoportok kialakításával is foglalkozni.

2.2.2. Időtartam becslés

Az egyik legalapvetőbb feladat a várható idő meghatározása, hiszen ez az egyik legmeghatározobban pontja a kerékpározásnak - mennyi időbe fog telni mire megtesz valaki adott távolságot? Ennek a kérdésnek a megválaszolására alapvetően a különböző regressziók alkalmasak, amik az út paramétereit 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.

A különböző időtartamok kiszámításához első lépésként 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.

2.2.3. Edzésterv

A felhasználók számára javasolt edzésterveknek több lehetséges megvalósítási módja is van. A legteljesebb kivitelezés a térképen jelölt konkrét útvonal, célul kitűzött teljesítmény és részidőkkal, nehézségi szinttel. Azonban a szakdolgozat során egy lényegesen egyszerűbb, alapvető módszer megvalósítása a cél, aminek alapját a korábban meghatározott nehézségi csoportok fogják jelenteni.

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 [5] 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

A Ridge regresszió [6][7] kitűnően alkalmas a Legkisebb négyzetek módszerének problémáinak való kiküszöbölésére. A módszer matematikailag:

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

ahol $\alpha \geq 0$ komplexitási paraméter. Ez a paraméter befolyásolja a csökkenés (*shrinkage*) mértékét: minél nagyobb az α annál meredekebb a csökkenés.

Érdemes használni mikor:

- a magyarázó változók (jellemzők) száma meghaladja a vizsgálatok (adatsorok) számát egy adathalmazban
- a magyarázó változók (jellemzők) között korreláció figyelhető meg, tehát mikor az adathalmaz multikollinearis

Regularizáció: a Ridge regresszió L2-es típusú regularizációt valósít meg, azaz büntető kifejezésként (*penalty term*) az $\alpha \|w\|_2^2$ formula kerül felhasználásra.

Ridge és Legkisebb négyzetek módszere: a legkisebb négyzetek módszere nem különböztet meg "fontos" és "kevésbé fontos" jellemzőket, így magas jellemzőszám és kevés adatsor esetén túlleszkedést eredményez, valamint nehezen kezel multikollinearis adathalmazt. A Ridge regresszió kiküszöböli ezeket a problémákat ugyanis pontosan elegendő torzítást (*bias*) használ ahhoz, hogy a becslések ésszerűen megbízhatóak legyenek.

Lasso

A LASSO regresszió [8][9] a *Least Absolute Shrinkage and Selection Operator* angol kiifejezés rövidítése. Ez egy lineáris modell amely ritka együtthatókat (*sparse coefficients*)

becsül. Hasznos lehet tömörített érzékelés (*compressed sensing*) esetén mivel a kevésbé magyarázó változót tartalmazó megoldások felé tendál, így effektíve csökkentve az eredményt befolyásoló jellemzők számát. Matematikailag:

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

ahol $\alpha \geq 0$ konstans együttható az L1 regularizációt befolyásoló, csökkenés mértékét jelző paraméter.

- $\alpha = 0$ esetén egyetlen jellemző sem kerül elvetésre, gyakorlatilag egy lineáris regresszió
- α növekedésével egyre több jellemző kerül elvetésre. Elméletileg $\alpha = \infty$ esetben minden jellemző elvetésre kerül
- α növekedésével nő az elfogultság (*bias*). Magas bias esetén alulilleszkedés figyelhető meg.
- α csökkenésével nő a variancia (*variance*). Magas variancia esetén túlilleszkedés figyelhető meg.

A Lasso regresszió tehát támogatja a minél egyszerűbb, ritka (*sparse*) modelleket. Jól használható multikollinearitás adathalmaz esetén illetve modell szelekció automatizálásához (pl.: magyarázó változók kiválasztása).

Random Forest

A Random Forest [10][11] modell egy additív típusú modell amely a jellemzők közötti összefüggések feltárásához döntési fákat kombinál egy végső modellbe. Formálisabban:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + \dots$$

ahol a g végleges modell az f_i alapmodellek összessége. minden f_i egy döntési fa. A Random Forest érdekessége hogy az összes alapmodell egymástól függetlenül, a rendelkezésre álló adat különböző részhalmazait felhasználva kerül kialakításra.

A módszer előnye a lineáris modellekkel szemben hogy képes a nem-lineáris összefüggéseket is megtalálni a magyarázó és a függő változók között.

2.3.2. Klaszterezés

A klaszterezés egy felügyelet nélküli gépi tanulási módszer, amelynek célja egy adathalmazt különálló csoportokra (klaszterekre) felbontani az adathalmazban fellelhető jellemzők alapján. A klaszterek kialakításakor a cél minden valamilyen hasonlóság megtalálása az egyes adatpontok között, így az adott szempontból hasonló adatok fognak egy klasztert alkotni. Hasonlóság meghatározására rengeteg különböző módszer létezik, két legnagyobb csoportja körül az első a tömörséget, távolságot veszi alapul a második pedig a pontok kapcsolódását vizsgálja. Mindkét csoportban rengeteg különböző megvalósítás létezik.

K-Means

A K-Means [13] eljárás az egyik legelterjedtebb, széles körben használt klaszterezési módszer, amely az adatpontokat távolságuk alapján csoportosítja. A módszer alapgondolata azonos szórású klaszterek kialakítása a tehetetlenség (*inertia*) vagy másképp a klasztereken belüli négyzetösszegek minimalizálásával:

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$

, ahol n az X adathalmaz darabszáma, C a c_j klasztereket tartalmazó halmaz és μ_j a c_j klaszter középpértéke, ami egyben a klaszter jellemzésére is szolgál. A μ_j pontot szokás középpontnak is nevezni, azonban ez az érték általában nem eleme az X halmaznak. A K-Means algoritmus az n mintát K diszjunkt klaszterre osztja, ahol K értékét előre definiálni kell.

A módszer jól teljesít nagy mennyiségű mintát tartalmazó adathalmazon azonban koránt sem tökéletes:

- feltételezi hogy a klaszterek konvexek és izotrópok, ami természetesen nem minden teljesül. Ebből kifolyólag rosszul teljesít hosszúkás vagy szabálytalan elrendezésű adaton
- a minimalizálásra kijelölt tehetetlenség nem normalizált mérőszám, annyit tudni róla hogy a kisebb értékek jobbak és nulla az optimális. Így az Euklideszi távolság használata miatt magas dimenziószámú adathalmaz esetén erősen ajánlott valamilyen dimenzió redukciós eljárás alkalmazása

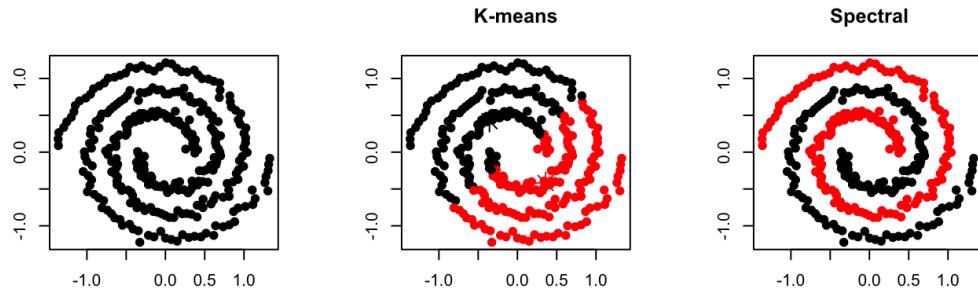
Mini-Batch K-Means

A Mini-Batch K-Means klaszterezési eljárás a K-Means egy variánsa, amely a számítási idő csökkentésének érdekében mini kötegeket (*batch*) használ, ugyanakkor igyekszik optimalizálni ugyanazt a célfüggvényt. Ezek a mini kötegek az adathalmaz véletlenszerűen választott részhalmazai, amelyek az eljárás során egységeként kezelve kerülnek hozzárendelésre a legközelebbi klaszter középponthoz. Más K-Means gyorsító eljárásokkal szemben a Mini-Batch előnye hogy általánosságban nagyon kicsit a eredmény romlása az alap K-Means-hez képest.

Spectral

A Spectral klaszterezési módszer [15] [16] a K-Means algoritmussal szemben nem a pontok távolsága alapján alakítja ki a klasztereket, hanem azok kapcsolatát veszi alapul: hiába esik közel egymáshoz két pont, amennyiben nem kapcsolódtnak nem kerülnek egy klaszterbe. A módszer az adathalmazt egy alacsony dimenziójú térré vetíti le és itt alakít ki egy ún. affinitási mátrixot. Ez az affinitási mátrix tárolja hogy mely pontok kapcsolódnak egymáshoz: ehhez a pontok páronkénti távolságát vizsgálja. Két pont között akkor áll fent kapcsolat ha távolságuk kisebb mint valamilyen ϵ érték.

A Spectral és a K-Means algoritmusok különbözőségét a 2.1. ábra szemlélteti. Látható hogy míg a K-Means gyakorlatilag egy vonal mentén kettészeli az adathalmazt, nem törődve a spirális alakzattal, addig a Spectral klaszterezés a két spirál vonal alapján alakítja ki a két klasztert.



2.1. ábra. K-Means és Spectral klaszterezés összehasonlítása [17]

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 mindenkor érzékeny terület, sok időt, energiát felemész és a kívánt eredmény ezek befektetésével sem mindenkor 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

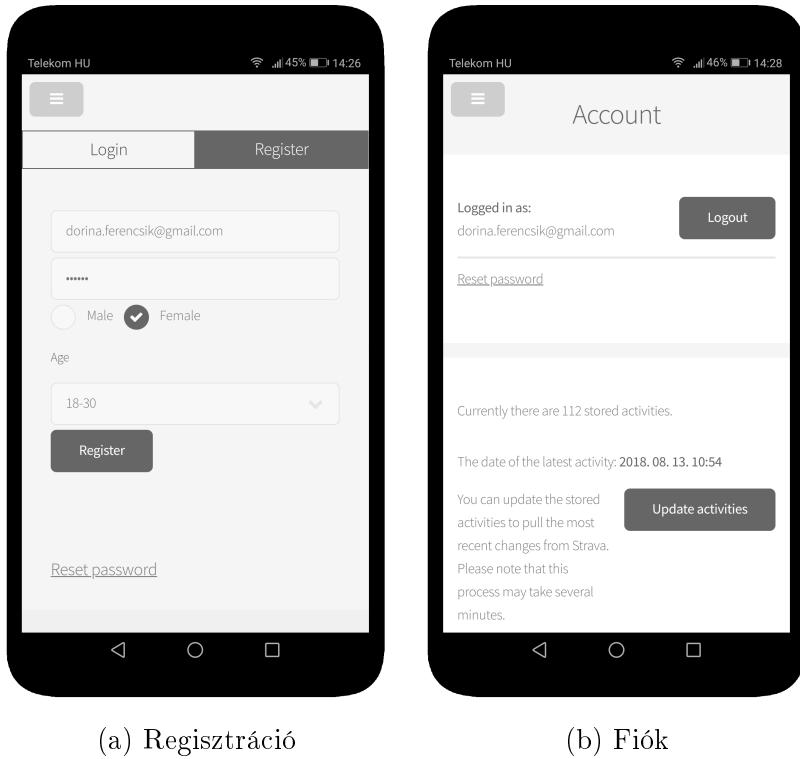
A Strava által gyűjtött és tárolt adatok elérésének alapját egy, a szakdolgozat keretein belül készített weboldal adja, amely a <https://cycling-with-ml.firebaseioapp.com> címen érhető el. Ezen oldal felkeresésével a felhasználók tájékozódhatnak a szakdolgozatról, az adatgyűjtés fontosságáról valamint alapvető funkciókat használhatnak. Az oldalon lehetőség van új felhasználó létrehozására és 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 weboldal az alábbi eszközök felhasználásával valósult meg:

JQuery

A JQuery [18] egy népszerű JavaScript könyvtár amely lehetőséget ad a HTML elemek könnyebb manipulálására, az események egyszerűbb kezelésére, animációk megvalósítására.

Firebase

A Firebase [19] 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 kezelní és a Firebase funkcióit weboldalakban és mobil alkalmazásokban is (mind Android és iOS) egyszerűen lehet használni.



(a) Regisztráció

(b) Fiók

2.2. ábra. Adatgyűjtő weboldal alapvető funkciói

A rendszer alapját egy Node.js szerver és egy JSON alapú adatbázis adja valamint elérhető egy általános célú tárhely tetszőleges file-ok tárolásához.

Firestore - Autentikáció

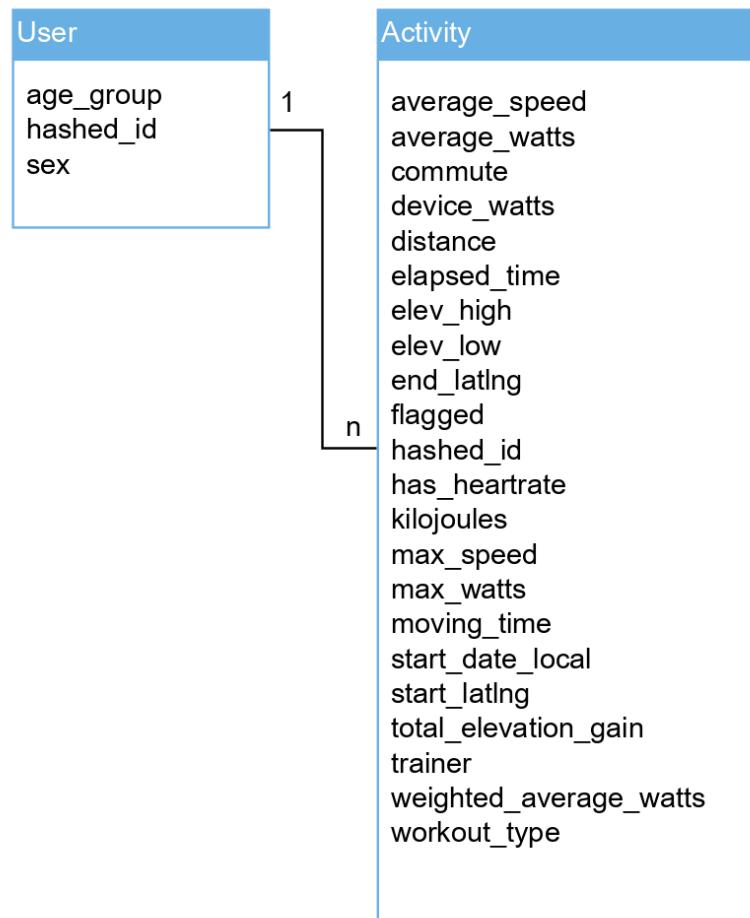
A szakdolgozathoz készített weboldal szempontjából a Firestore egyik legfontosabb eleme az autentikációs rendszer. A Firebase felületén való engedélyezés után a weboldal kódbázisában gyorsan és biztonságosan megvalósítható az új felhasználók regisztrálása, bejelentkeztetése. Az email alapú regisztráláson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

Firestore - 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 kliens oldalon történő 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 azonosítójá 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 jellemzi 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 valamit a felhasználókhöz kapcsolódó adatok szerkezete a 2.3. ábrán látható, leírásuk a következőben kerül részletezésre.



2.3. ábra. Adatstruktúra

Jellemzők magyarázata:

- **age_group:** felhasználó korcsoportja (0: 18-30; 1: 31-45; 2: 45+)
- **hashed_id:** felhasználó azonosító hash-elt változata
- **sex:** felhasználó neme (male / female, férfi / nő)
- **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' felcím-kézni, ezzel megjelölve hogy nem egy verseny vagy túra adat, hanem mindennap, 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; -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)
- **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)
- **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 (logikai)
- **weighted_average_watts:** súlyozott átlagos erőkifejtés az útvonal során. Ha nem biztosított az értéke: -1.
- **workout_type:** tevékenység típusa, lehet edzés vagy verseny (egész szám, ha nincs megadva akkor -1. Lehetséges értékei: 10 - normál kerékpározás; 11 - verseny; 12 - edzés)

3. fejezet

Adatfeldolgozás és gépi tanulás

Az Elméleti kifejtés során meghatározott célokat különböző gépi tanulási módszerekkel lehet megoldani amik eltérő eredménnyel, pontossággal fognak működni. A nehézségi osztályok meghatározására elsősorban a 2.3.2. pontban részletezett klaszterezési módszerek használhatóak.

3.1. Adathalmaz előkészítés

A fejlesztés legelső lépése a megfelelő adatstruktúrára kialakítása és az adathalmaz tisztítása, ezzel megkönnyítve a későbbi feldolgozást valamint növelte a gépi tanulási algoritmusok pontosságát.

A struktúrát meghatározza az adattárolásra használt csv fájl szerkezete, ahol minden sor egy-egy megtett útvonalat jelöl az oszlopnevekben meghatározott jellemzők alapján. Ennek a fájlnak a betöltésre a Pandas [20] nevű Python csomag szolgál, az adatok futás közbeni tárolására a csomagnak a DataFrame nevű adatstruktúrája kerül felhasználásra. A DataFrame kényelmes, ember közeli adatkezelést tesz lehetővé, az adatok oszlopnev és index alapján is könnyen elérhetőek, szűrhetőek.

3.1.1. 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.

A megtaláláshoz első lépés az adatok áttekintése: NULL értékek vizsgálata, egyes jellemzők eloszlásának megtekintése, minimum, maximum és a kvantilisek összehasonlítása. Ehhez nyújt segítséget a 3.1. táblázat amely a numerikus oszlopokról készült általános jellemzést összegzi. A táblázat oszlopainak a jelentése:

- Count: jellemző nem NULL értékeinek darabszáma
- Mean: jellemző átlaga
- STD (Standard Deviation): jellemző szórása

Néhány alapvető gond már a táblázat alapján is felfedezhető: sok jellemző tartalmaz NULL értékeket (*average_watts*, *kilojoules* stb.), máshol pedig olyan értékek szerepelnek amelyek érvénytelenek, értelmezhetetlenek az adott jellemzőre nézve (pl.: *distance*

	Count	Mean	STD	Min	0.25	0.50	0.75	Max
age_group	10014	1.01	0.48	0.0	1.0	1.0	1.0	2.0
average_speed	10014	24.25	653.31	0.0	4.96	5.73	6.47	36100.0
average_watts	1848	163.0	49.51	0.0	139.10	161.85	180.85	482.6
distance	10014	26153.09	29465.23	0.0	7444.83	13751.25	34587.2	436806.0
elapsed_time	10014	5845.88	19296.2	0.0	1648.00	3020.0	7343.0	1806211.0
elev_high	9845	282.1	272.66	-71.2	162.2	229.4	296.4	12103.0
elev_low	9844	131.99	68.68	-500.0	103.1	130.0	148.3	1484.0
kilojoules	1765	1295.76	756.06	0.0	728.7	1236.7	1745.4	5928.8
max_speed	10014	12.14	3.99	0.0	9.6	11.6	14.48	122.7
max_watts	546	630.95	244.34	115.0	502.0	603.0	708.0	2428.0
moving_time	10014	4543.18	5016.52	0.0	1442.0	2451.0	6109.0	90983.0
total_elevation_gain	10014	278.7	428.88	0.0	27.5	85.2	361.8	4416.8
weighted_average_watts	546	188.86	30.35	5.0	176.25	195.0	208.75	246.0
workout_type	4039	9.74	1.85	0.0	10.0	10.0	10.0	12.0

3.1. táblázat. Nyers adathalmaz numerikus oszlopainak leírása

legkisebb értéke 0 méter; *average_speed* legnagyobb értéke 36100 m/s (129960 km/h)). Az ehhez hasonló hibás adatok megtalálása és valamilyen jellegű javítása kritikus feladat, részletezésük a továbbiakban található.

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 eljárás 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 tényleges értéket tartalmaz.

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 Igaz érték tehát valójában kevesebb mint 1.0% használható
- device_watts : 19.56% hasznos értéket tartalmaz
- flagged : 19.56% hasznos értéket tartalmaz
- has_heartrate : 19.56% hasznos értéket tartalmaz
- kilojoules : 17.63% hasznos értéket tartalmaz
- max_watts 5.54% hasznos értéket tartalmaz
- weighted_average_watts : 5.45% hasznos értéket tartalmaz
- 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 a jellemző funkcióját veszti, konstans érték
- start_latlng : 18.34% hasznos értéket tartalmaz
- end_latlng : 18.34% hasznos értéket tartalmaz

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 vizsgálva egy F jellemzőre $x \in F$ érték kiugrónak tekinthető amennyiben

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

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

Az alábbiakban a különböző jellemzők kiugró érték vizsgálatának részletes leírása található.

Átlagsebesség (average_speed): 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 lehetlen kerékpárral, eredetük az adatsorok megvizsgálása után leszűkíthető a mozgási idő (*moving_time*) jellemző mérésének a hiányára / hibájára. Az átlagsebesség 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.

Távolság (distance): 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 érték 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 későbbiekkorán érdemes lehet a hasonlóan hosszú utakat külön kezelní.

Legmagasabb tengerszint feletti pont (elev_high): 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.

Legnagyobb sebesség (max_speed) 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/h. Ezek olyan adatok amelyek az adathalmazra nézve kiugró értékűek, sok közülük lehetetlen / értelmezhetetlen. Ezen adatsorok eldobása után az adathalmaz 9725 útvonalat tartalmaz.

Mozgási idő (moving_time) 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 nem kerülnek törlésre.

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:

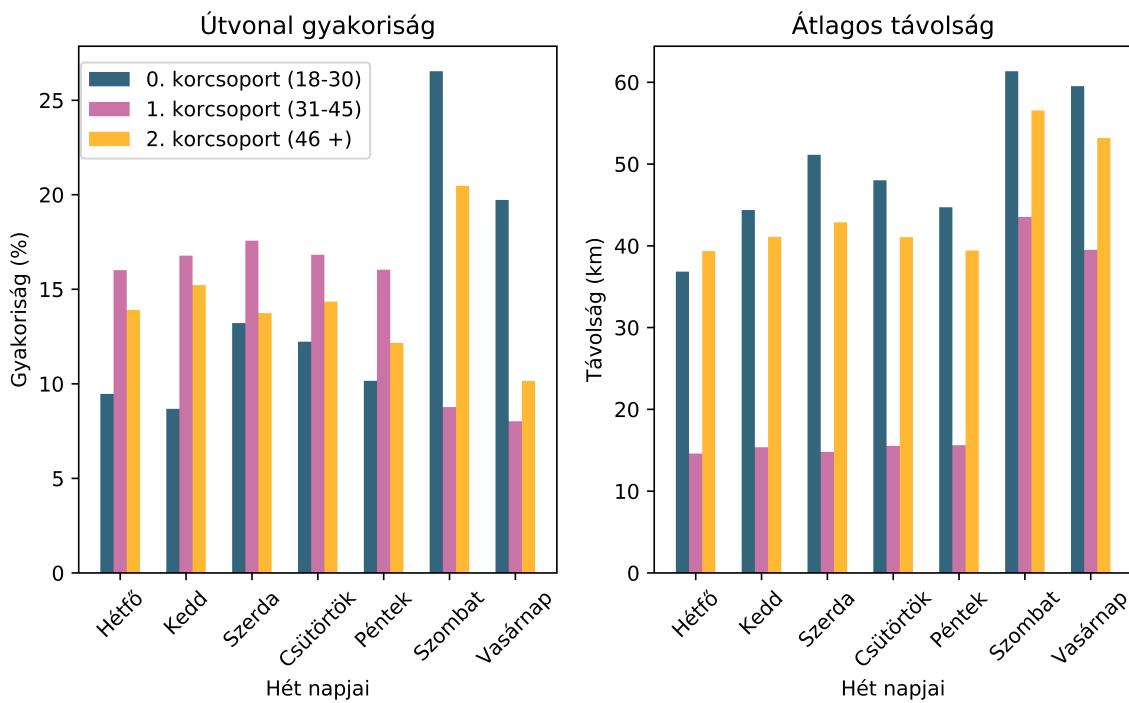
- az átlagsebesség kisebb mint 2 m/s (7.2 km/h)
- a távolság kisebb mint 100 méter (néhány száz méter esetén előfordulhat hogy rövid sprinteket tettek meg a sportolók)
- a mozgás típus (*workout_type*) jellemző értéke 0 vagy 4
- a legmagasabb vagy legalacsonyabb tengerszint feletti magasság hiányzó adat

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

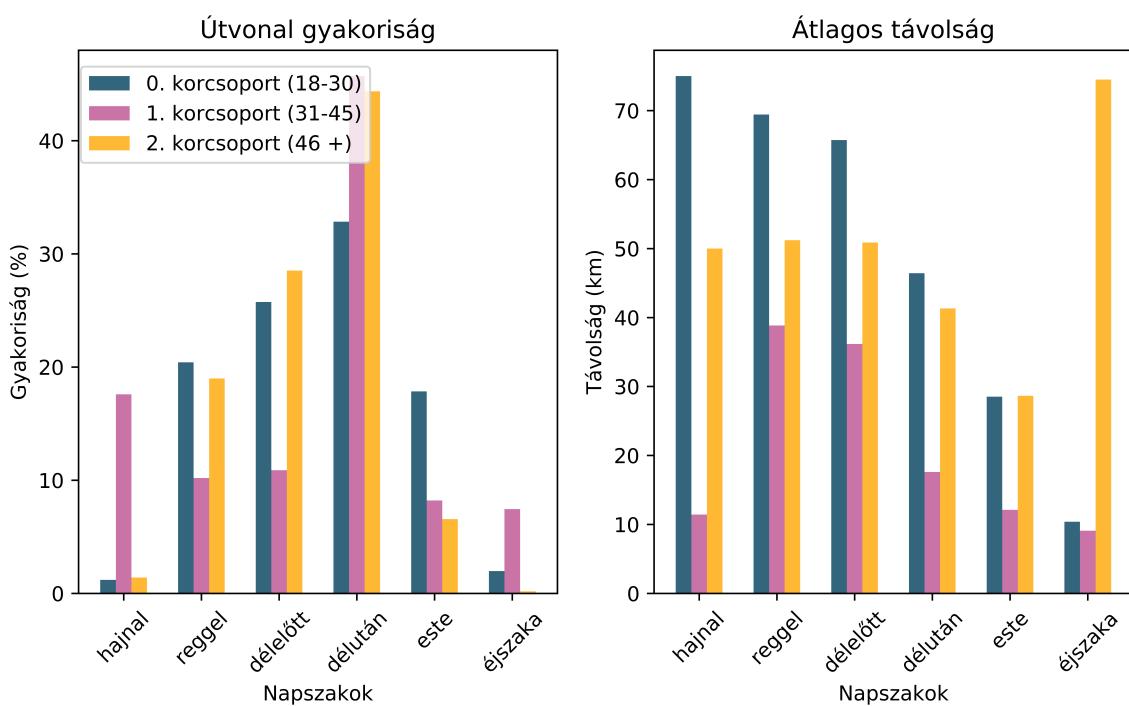
A fent definiált szabályok alkalmazása után az adathalmaz összesen 9474 útvonalat tartalmaz.

3.1.2. Adatok áttekintése

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



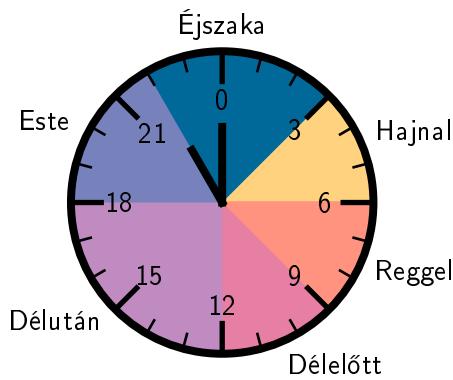
3.1. ábra. Átlagos távolság és útvonal gyakoriság a hét napjain



3.2. ábra. Átlagos távolság és útvonal gyakoriság napszakok szerint

A 3.1. ábrán két grafikon jellemzi az útvonalakat a hét napjai szerint. A bal oldali gráf az útvonalak darabszámának eloszlását jelzi, a jobb oldali pedig az átlagosan megtett távolságot, mindenkor esetben korcsoportok szerint lebontva. Érdekesség hogy az 1. korcsoportba tartozó sportolók által megtett útvonalak nagyjából 80%-a hétköznapra esik, azonban ezen útvonalak átlag távolsága mindenkor 15 km körül mozog szemben a hétvégi kerékpározások átlagos 40 kilométerével. Természetesen minden korosztály esetén a hétvégén megemelkedik az átlagosan megtett távolság hiszen mindenkor mindenkor több ideje van nagyobb túrák teljesítésére. Hét közben nagy eltérés figyelhető meg az átlagos távolság esetében: az 1. korcsoportba tartozó kerékpározók messze elmaradnak a másik két korcsoportba tartozók mögött. Ennek egy lehetséges kiváltó oka hogy az 1. korcsoportban valószínűleg mindenki dolgozik, esetleg építkezik, családra koncentrál így érthetően kevesebb idejük van hosszú túráakra, valószínűleg munkába járás, közlekedés céljából kerékpároznak inkább hétköznapokon.

A 3.2. ábrán látható két grafikon az útvonalak gyakoriságát és az átlagosan megtett távolságot napszakokra lebontva jeleníti meg. A napszakok kialakítása szubjektív módon, a 3.3. ábrán található 24 órás lebontás alapján történt.



3.3. ábra. Napszakok kialakítása

3.1.3. 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 minden érdemes alkalmazni.

A szakdolgozathoz használt adathalmaz esetén 3+1 jellemző kerül one-hot encode-olásra: age_group (korcsoport), workout_type, 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, $+\infty$) korcsoportokba

való tartozást jelenti. One-hot encoding során az eredeti jellemzőből 3 oszlop készül, végül az eredeti törlésre kerül az adathalmazból.

3.1. 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_type: az aktivitás típusát jellemzi, értékei egy 3 elemű halmazból kerülhetnek ki: {10, 11, 12}, ahol 10: általános kerékpározás / jelöletlen; 11: verseny, 12: edzés.

3.2. Program részlet.

A *workout_type* kategorikus jellemző, így a számítások előtt One-hot encoding alkalmazásával lebontásra került az alábbi kód segítségével.

```

1 workoutTypeOneHot = pandas.get_dummies(rawData[ 'workout_type' ] ,
2                                         prefix='workout_type')
3 rawData.drop(columns='workout_type' , inplace=True)
4 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

3.3. Program részlet.

Az aktivitás kezdetének dátumából kinyerhető hogy a hétfély napján történt az aktivitás. Ezen információ 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)
```

3.4. Program részlet.

Az aktivitás kezdetének időpontjából kinyerhető hogy melyik napszakban kezdődött az aktivitás. Ezen információ 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 3.3. ábra szerint

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 átalakítása 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: átlagsebesség az útvonal során (méter / másodperc)
- distance: összesen megtett távolság (méter)
- elapsed_time: összesen eltelt idő (másodperc)
- elev_high: útvonal legmagasabb pontja (méter)
- elev_low: útvonal legalacsonyabb pontja (méter)
- hashed_id: sportoló hashelt azonosítója
- max_speed: legnagyobb elérte sebesség (méter / másodperc)
- moving_time: mozgással töltött idő (másodperc)
- total_elevation_gain: összesen megtett emelkedés (méter)
- age_0.0: 0. korcsoportba való tartozást jelöli (bináris)
- age_1.0: 1. korcsoportba való tartozást jelöli (bináris)
- age_2.0: 2. korcsoportba való tartozást jelöli (bináris)
- trainer_onehot: útvonal megtétele gépen történt e (bináris)
- workout_type_10.0: 10 típusú útvonal (bináris)
- workout_type_11.0: 11 típusú útvonal (bináris)
- workout_type_12.0: 12 típusú útvonal (bináris)
- weekday_Monday: hétfői aktivitás (bináris)
- weekday_Tuesday: keddi aktivitás (bináris)
- weekday_Wednesday: szerda aktivitás (bináris)
- weekday_Thursday: csütörtöki aktivitás (bináris)
- weekday_Friday: pénteki aktivitás (bináris)
- weekday_Saturday: szombati aktivitás (bináris)

- weekday_Sunday: vasárnapi aktivitás (bináris)
- daypart_dawn: hajnali indulás (bináris)
- daypart_morning: reggeli indulás (bináris)
- daypart_forenoon: délelőtti indulás (bináris)
- daypart_afternoon: délutáni indulás (bináris)
- daypart_evening: esti indulás (bináris)
- daypart_night: éjszakai indulás (bináris)

Az adathalmaz ezen a ponton mentésre kerül a későbbi felhasználás megkönnyítésének céljából. A tisztított adatot a *cleaned_data_{date}.csv* fájl tartalmazza.

A tisztított adathalmaz numerikus oszlopainak összefoglalása a 3.2. táblázatban található.

	count	mean	std	min	0.25	0.50	0.75	max
average_speed	9603	5.76	1.22	2.03	4.99	5.74	6.46	12.48
distance	9603	26140.85	29199.25	161.0	7451.3	13702.2	34599.1	436806.0
elapsed_time	9603	5817.39	19632.69	61.0	1645.0	2998.0	7338.5	1806211.0
elev_high	9603	270.73	166.61	-71.2	162.3	229.4	290.5	956.0
elev_low	9603	129.92	47.88	-134.4	103.2	130.2	148.3	809.6
max_speed	9603	12.22	3.24	2.3	9.7	11.6	14.4	22.0
moving_time	9603	4509.68	4948.21	61.0	1440.0	2415.0	6099.0	90983.0
total_elevation_gain	9603	274.47	410.03	0.0	29.8	87.0	359.45	3896.8

3.2. táblázat. Tisztított adathalmaz numerikus oszlopai

3.2. Regresszió

Az adathalmaz egy aktivitásra tekintve két különböző időtartamot különböztet meg: a mozgással töltött (*moving_time*) és az összesen eltelt (*elapsed_time*) időt, mindenkorrel másodpercben mérve. Ennek a két időtartamnak a becslése a 2.3.1. alfejezetben kifejtett regressziók felhasználásával történik.

A kétféle időtartam alapvetően külön kezelve, egyenként kerül becslésre.

3.2.1. Paraméter hangolás

A gépi tanulási algoritmusok tanításának eredményessége kis mértékben javítható az adott eljárás paramétereinek a használt adathalmazhoz való igazításával. Az algoritmusok a scikit-learn megvalósításában rendelkeznek alapparaméterekkel, tehát egy alapvető eredmény ezek explicit beállítása nélkül is számítható, azonban a megfelelő paraméterek kiválasztása akár néhány százalékos javulást is eredményezhet.

Ridge regresszió:

ez az algoritmus 2 fő paraméterrel finomhangolható: $\alpha \geq 0$ és $solver \in \{\text{auto}, \text{svd}, \text{cholesky}, \text{lsqr}, \text{sparse_cg}, \text{sag}, \text{saga}\}$. Az α paraméter a regularizáció mértékét befolyásolja a különböző $solver$ -ek pedig a számítási rutint befolyásolják (lásd.: dokumentáció). Az algoritmus optimalizálása során $\alpha \in \{0.0001, 0.0003, 0.001, 0.003, 0.01, 0.1, 0.3, 0.35, 0.6\}$ értékek kerültek figyelembe vételre.

Lasso regresszió:

a modell finomhangolása elsősorban az $\alpha \geq 0$ paraméter változtatásával történik. A scikit-learn implementáció lehetőséget ad egyéb opciók beállítására azonban ezek nem befolyásolják nagyban a modell működését így használatuk mellőzésre kerül a szakdolgozat során. Az optimalizálás során $\alpha \in \{0.0001, 0.0003, 0.001, 0.003, 0.01, 0.1, 0.3, 0.35, 0.6\}$ értékek kerültek figyelembe vételre.

Random Forest regresszió:

ez a döntési fákon alapuló eljárás lényegesen mélyebben személyre szabható a paraméterei révén mint a Ridge és a Lasso regresszió. Optimalizálása [23] során a következő paraméterek kerültek felhasználásra:

- $n_estimators \in \{200, 500, 1000\}$: fák száma
- $max_features \in \{\text{auto}, \text{sqrt}\}$: a legjobb felosztás megtalálásához felhasználendő jellemzők darabszáma
- $min_samples_leaf \in \{10, 30, 60\}$: egy levél legalább hány mintát tartalmazzon

A Random Forest egyéb paraméterei is befolyásolhatják a tanulás eredményességét, a becslések pontosságát.

A tanítás előtt elő kell készíteni az adattisztítást eredményeként kapott ömlesztett adathalmazt. Ennek első lépése a magyarázó X és a függő y halmazok kialakítása. Ezeknek változniuk kell annak függvényében hogy a mozgással töltött időt vagy az összesen eltelt idő becslése e a cél aktuálisan. Azonban miután kialakításra kerül a két adathalmaz fontos hogy a regressziós modellek tanítása előtt standardizálásra kerüljenek. Ennek fontossága akkor mutatkozik meg igazán mikor a különböző jellemzők nagyon eltérő intervallumokba esnek, például a $average_speed$ 2.03 és 12.478 értékek között mozog, míg a $distance$ jellemző 161 és 436806 között. A standardizálás elvégzésére a szakdolgozat során a scikit-learn csomag által biztosított StandardScaler függvény szolgál.

3.5. Program részlet. Adathalmaz általános standardizálása

```
1 from sklearn.preprocessing import StandardScaler
2 def getStandardized(X):
3     X = dataset.copy()
4
5     names = X.columns
```

```

6     scaler = StandardScaler()
7
8     scaledData = scaler.fit_transform(X)
9     scaledData = pandas.DataFrame(scaledData, columns=names)
10    return scaledData

```

3.6. 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. A tanító és teszt adathalmazok kialakítására a scikit-learn csomag által biztosított *train_test_split* függvény szolgál.

```

1 from sklearn.model_selection import train_test_split
2
3 scaledY = scaledData['{target_feature}']
4 scaledData.drop(columns=['{target_feature}'], inplace=True)
5
6 trainX, testX, trainy, testy = train_test_split(scaledData, scaledY,
7                                                 random_state=42)

```

3.2.2. Mozgási idő

A mozgási idő becsléséhez három különböző adathalmaz kerül kialakításra:

- *all*: az *X* halmaz a *moving_time* az *elapsed_time* és az *average_speed* oszlopok kivételével minden jellemzőt tartalmaz
- *reduced*: az *X* halmazban csupán a következő jellemzők találhatóak meg: *age_0.0*, *age_1.0*, *age_2.0*, *distance*, *elev_high*, *elev_low*, *hashed_id*, *total_elevation_gain*, *trainer_onehot*
- *base*: az *X* halmaz kizárolag az útvonal fizikai jellemzőit tartalmazza: *distance*, *elev_high*, *elev_low*, *total_elevation_gain*, *trainer_onehot*, *text*.

Mindkét adathalmaz esetén *y* csak a *moving_time* jellemzőt tartalmazza és természetesen megtörténik az adatok standardizálása valamint tanító és tesztelő részhalmazokra való bontása a 3.5. és a 3.6. programrészletek segítségével.

Eredmények

A Ridge és Lasso regresszió eredményei a 3.3. táblázatban kerülnek összefoglalásra. Mint látható a két különböző modell szinte identikus pontosságot ad a teszt adathalmazokra, minimális csökkenéssel a *reduced X* esetén.

A Ridge és Lasso regressziók működése nagyon közel áll egymáshoz, csupán regularizáció tekintetében különböznek, így nem meglepő hogy a két módszer hasonló pontossággal ad becsléseket.

A Random Forest regresszió eredményei a 3.4. táblázatban kerülnek összefoglalásra. Ez a döntési fákon alapuló eljárás alapjaiban különbözik a Ridge és Lasso regresszióktól azonban a szakdolgozatban használt adathalmazon közel azonos pontossággal teljesít. Általában 2-3%-al ad rosszabb eredményeket mint a Ridge és Lasso regressziók.

Modell	X	y	Pontosság	α	Solver
Ridge	all	moving_time	0.935588	0.3	saga
Ridge	reduced	moving_time	0.933802	0.1	sag
Ridge	base	moving_time	0.920782	0.001	sag
Lasso	all	moving_time	0.935583	0.0001	–
Lasso	reduced	moving_time	0.933784	0.0001	–
Lasso	base	moving_time	0.920763	0.0001	–

3.3. táblázat. Ridge és Lasso eredmények moving_time jellemző becslésére

X	Pontosság	max_features	min_samples_leaf	n_estimators
all	0.910675	auto	10	500
reduced	0.908003	auto	10	500
base	0.894923	auto	10	200

3.4. táblázat. Random Forest regresszió eredmények moving_time jellemző becslésére

Összefoglalás

A mozgási idő (*moving_time*) jellemző szorosan összefügg az útvonalak fizikai tulajdonságaival ezért egyszerű regressziókkal is nagy pontossággal becsülhető. Azonban a 3.3. és a 3.4. táblázatokban látható hogy csak az útvonal fizikai adatai alapján (*base* adathalmaz) elérte pontosság javítható egyéb, a sportolóhoz és időponthoz kapcsolható adatok segítségével. Ez alapján egyéb adatok segítségével (sportolóhoz, időponthoz kapcsolódó) a modellek finomhangolhatóak, nagyobb pontosság érhető el.

3.2.3. Összesen eltelt idő

A 3.5. és a 3.6. program részletek segítségével 7 különböző adathalmaz kerül kialakításra. A cél mindenkor az *elapsed_time* jellemző becslése amelyet egyik X sem tartalmaz

- **all:** X minden jellemzőt tartalmaz
- **reduced:** X csökkentett számú jellemzőt tartalmaz, név szerint a következőket: *age_0.0*, *age_1.0*, *age_2.0*, *distance*, *elev_high*, *elev_low*, *hashed_id*, *total_elevation_gain*, *moving_time*, *trainer_onehot*
- **ageNull:** X és y is csak a 0 korcsoportba (18-30) tartozó sportolók adatait tartalmazza
- **ageOne:** X és y is csak az 1 korcsoportba (31-45) tartozó sportolók adatait tartalmazza
- **ageTwo:** X és y is csak a 2 korcsoportba (46+) tartozó sportolók adatait tartalmazza
- **distanceSmall:** X csak olyan útvonalakat tartalmaz ahol a távolság kisebb mint 50 km

- **distanceBig:** X csak olyan útvonalakat tartalmaz ahol a távolság nagyobb mint 50 km
- **user:** a legtöbb útvonallal rendelkező felhasználó adatait foglalja magába

Eredmények

A Ridge és Lasso regressziók eredményeit a 3.5. táblázat tartalmazza. Látható hogy az adat különböző részhalmazaira nagyon eltérő eredmények születtek, néhány esetben kiemelkedően rossz és meglepően jó pontosság is megfigyelhető. A mozgási idő kiegyensúlyozottságával szemben az eltelt időre a Lasso regresszió mindig jobban teljesít mint a Ridge. Általában csak 1% körüli a teljesítmény különbség, de például a nagy távolságot tartalmazó adatok esetén (*distanceBig*) 8%-os különbség is megfigyelhető.

X	Modell	Pontosság	α	Solver
all	Ridge	0.722601	0.35	lsqr
all	Lasso	0.733696	0.01	–
reduced	Ridge	0.741402	0.6	saga
reduced	Lasso	0.742481	0.001	–
ageNull	Ridge	0.922961	0.003	saga
ageNull	Lasso	0.923146	0.003	–
ageOne	Ridge	0.756610	0.0	auto
ageOne	Lasso	0.774819	0.01	–
ageTwo	Ridge	0.914806	0.6	saga
ageTwo	Lasso	0.917202	0.003	–
distanceSmall	Ridge	0.325245	0.6	sag
distanceSmall	Lasso	0.384225	0.01	–
distanceBig	Ridge	0.823487	0.6	saga
distanceBig	Lasso	0.908215	0.1	–
user	Ridge	0.184641	0.6	saga
user	Lasso	0.184506	0.6	–

3.5. táblázat. Ridge és Lasso eredmények elapsed_time becsléséhez

A Random Forest regresszió eredményei a 3.6. táblázat tartalmazza. A pontosságot összehasonlítva a 3.5. táblázatban lévő Ridge és Lasso regressziók pontosságával meglepő módon a Random Forest regresszió az esetek nagy részében jobban teljesít. Kivélelt a 0. és 2. korcsoportot (*ageNull* és *ageTwo*) valamint a nagy távolságú útvonalakat (*distanceBig*) tartalmazó adathalmazok képeznek. A legnagyobb eltérés a felhasználói szintű adat (*user*) esetén figyelhető meg: ez az adathalmaz összesen 2985 útvonalat tartalmaz (ezzel a teljes adathalmaz majdnem harmadát adva).

X	Pontosság	max_features	min_samples_leaf	n_estimators
all	0.813672	sqrt	10	500
reduced	0.812520	sqrt	10	500
ageNull	0.901840	auto	10	200
ageOne	0.779302	sqrt	10	1000
ageTwo	0.772750	auto	30	500
distanceSmall	0.778130	sqrt	10	1000
distanceBig	0.902468	auto	10	500
user	0.839805	auto	10	200

3.6. táblázat. Random Forest regresszió eredmények elapsed_time jellemző becslésére

Összefoglalás

Az összesen eltelt idő ugyan összefügg az útvonalak fizikai adataival de nagy mértékben befolyásolják egyéb tényezők is. Így ennek a jellemzőnek a becslése lényegesen nehézkesebb mint a mozgással töltött időé. A eredmények alapján a kisebb, specifikusabb csoportokat tartalmazó adathalmazokra pontosabb becslések születtek. Például a korcsoport szerint lebontott adatok minden esetben nagyobb pontosságot eredményeztek mint az általános *all* adathalmaz. Érdekes módon a *reduced X* használatakor mind a Ridge mind a Lasso regresszió eredményesebb volt mint az *all* adathalmaz esetében. A legnagyobb kiingások a távolság szerint lebontott *X*-ek esetében található meg, valamint itt kell keresni a probléma magyarázatát is. Az 50 km-nél kettévágott (*distance_small* és *distance_big*) adatra készült becslések eredményei jól mutatják hogy a rövid útvonalaknál túl kevés az összefüggés ahhoz hogy pontos, használható modell készüljön belőle. Ennek oka hogy rövid távolság esetén egy specifikus felhasználó ismétlődő útvonalaira vetítve is változik az összesen eltelt idő, olyan befolyásoló tényezők hatására amiket az összegyűjtött adathalmaz nem tartalmaz, esetleg nem is mérhetők. Ilyen tényezők lehetnek városi utak esetén a közlekedési lámpák, forgalom, néhány perces megállások felfrissülés, telefonálás vagy akár egy gyors bevásárlás miatt, és ezeken kívül természetesen rengeteg más. Hosszútávú útvonalak esetén azonban ezek közelebb eshetnek egymáshoz, általánosabb a kerékpározók pihenési ideje, az adathalmaz nem tartalmaz akkora variációkat mint a rövidebb útvonalak esetén. Ennek következtében a különböző regressziós módszerek sokkal jobban illeszkednek a hosszabb útvonalak esetén, jobb becslésekkel adják a teszt adathalmaz elemeire.

3.3. Klaszterezés

A gyűjtött adathalmaz nem tartalmaz nehézségre vonatkozó osztály címkéket, így felügyelet nélküli klaszterezés segítségével kerülnek kialakításra különböző csoportok. A klaszterezési eljárások különböző szabályok alapján csoportosítják az eredményeket, némelyik eljárásnál az eredmények minimálisan eltérhetnek a többszöri futtatáskor, aminek oka hogy a klaszter (csoport) középpontok véletlenszerűen inicializálódnak és előfordulhat hogy egy lokális minimumhoz konvergál az algoritmus.

A nehézségi csoportok meghatározásához az adathalmaz korcsoportok szerint kerül felbontásra és ezeknek a részhalmazoknak csupán két jellemzője került felhasználásra:

az átlagos sebesség (*average_speed*) és a távolság (*distance*). Egy útvonalnak ez a két legalapvetőbb, atomi tulajdonsága, ami végső soron szinte minden másra kihatással van. Ezek a jellemzők minden korcsoport esetén a 2.3.2. alfejezetben kifejtett K-Means, MiniBatch K-Means és Spectral klaszterezési módszerrel kerültek feldolgozásra. A cél 5 klaszter meghatározása, amelyek elkülönítésével és tulajdonságaik körbehatárolásával egy egyszerű edzésterv javaslatot adó rendszer készíthető el.

A szakdolgozat során kidolgozott edzés tervező módszer egy nagyon alapvető számítás eredménye. A cél az egyes korcsoportokra különböző típusokat meghatározni és a típusokon belül eltérő nehézségű javaslatokat adni a felhasználók számára. Ennek megvalósításához a klaszterezési módszerekkel meghatározott típusokra külön - külön kerül kiszámításra az adott típusba tartozó adatok 25, 50 és 70%-a, ezzel 3 nehézségi fokozatot meghatározva. Továbbá meghatározásra kerül a várható időtartam a kapott távolság és átlagos sebesség alapján. Ennek a módszernek a megvalósítása a 3.7. programrészletben található.

3.7. Program részlet. Edzés javaslat számítása korcsoport (*ageGroup*), klaszterezési algoritmus (*clustering*), edzés típusa (*trainType*) és nehézsége alapján (*trainDifficulty*).

```

1 def getTrainingSuggestions( self , ageGroup , clustering , trainType ,
2                               trainDifficulty ):
3     if ageGroup == 0:
4         data = self .ageNullInverted
5     elif ageGroup == 1:
6         data = self .ageOneInverted
7     else :
8         data = self .ageTwoInverted
9
10    suggestion = data[ data[ clustering ] ==
11                           trainType ]. quantile( trainDifficulty )
12    # plotting suggestion

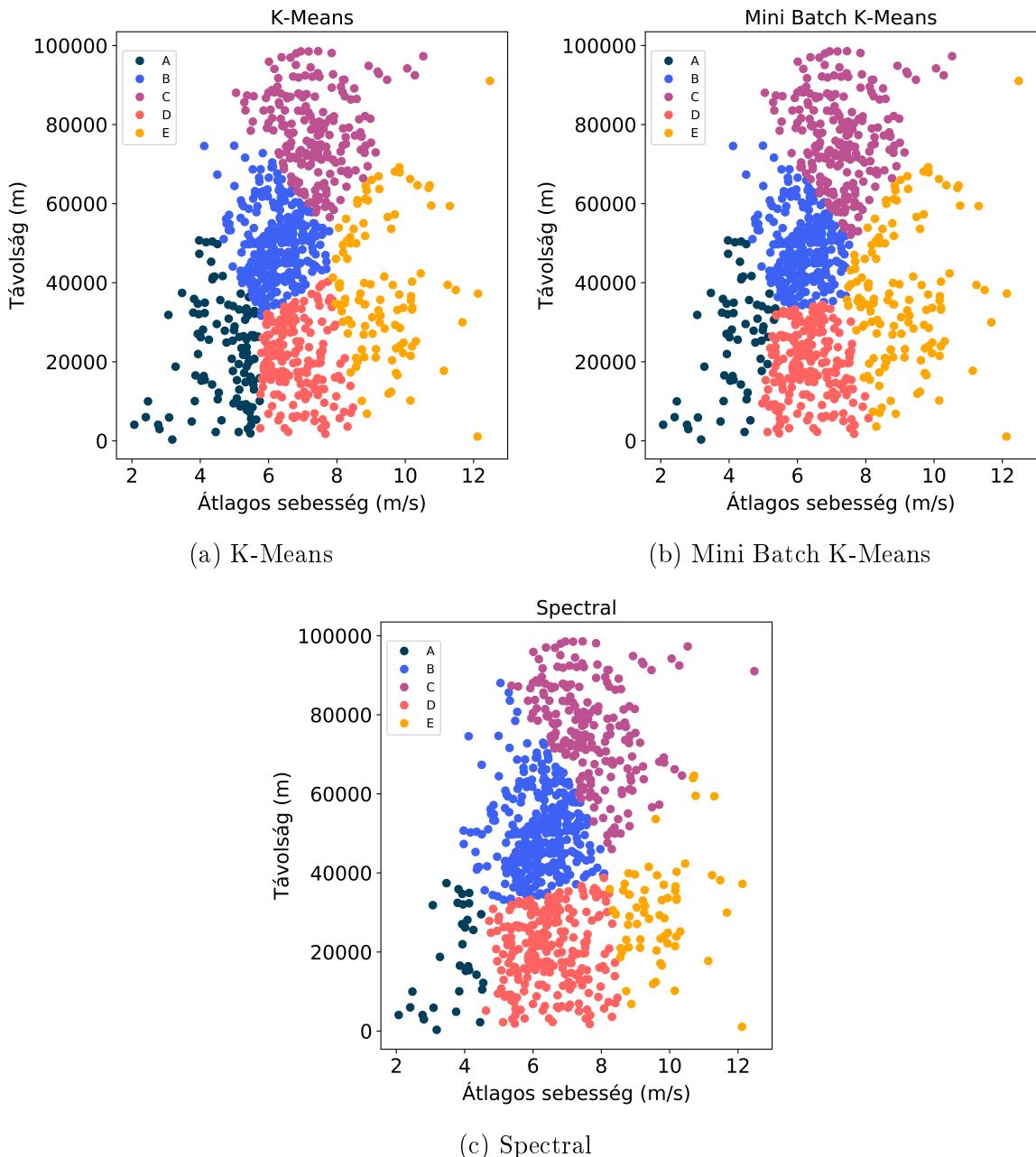
```

3.3.1. 0. korcsoport

A részhalmaz összesen 914 útvonalat tartalmaz. A különböző klaszterezési eljárások eredményei a 3.4. ábrán láthatóak. A kialakított klaszterek elhelyezkedése nagy vonalakban megegyezik, általánosan az alábbi módon lehetne jellemzni őket:

- A : normális és nagy távolság - normális átlagsebesség
- B : nagy távolság - normális és magas átlagsebesség
- C : extrém nagy távolság - magas és extrém magas átlagsebesség
- D : normális távolság - magas átlagsebesség
- E : normális és nagy távolság - extrém magas átlagsebesség

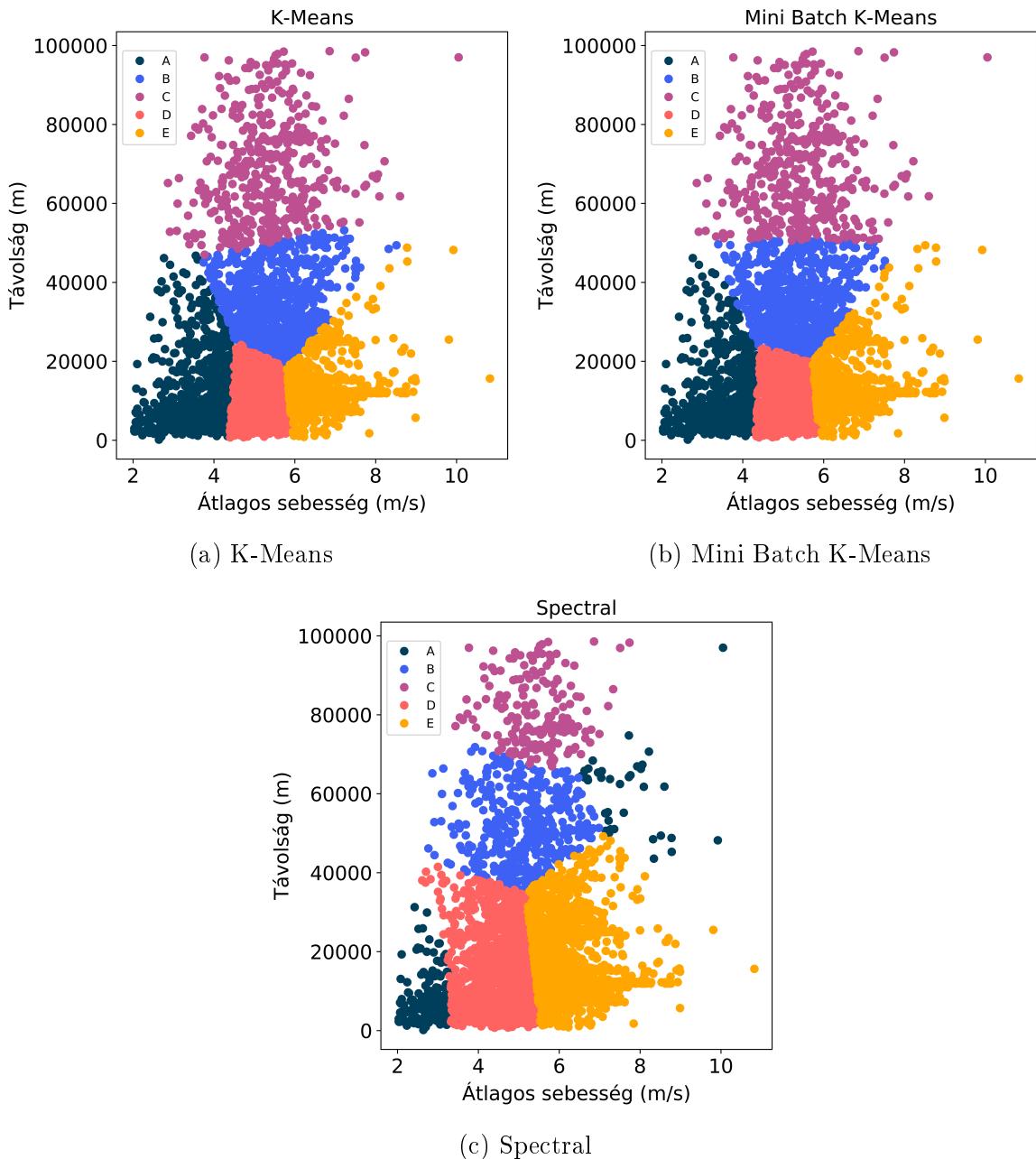
A K-Means és a Mini Batch K-Means legnagyobb eltérése az A és B-D klaszterek határán figyelhető meg: a Mini Batch esetén a B-D klaszterek kisebb átlagos sebességű adatokat is tartalmaznak, a klaszter határok közelebb esnek az y tengelyhez. Így az A



3.4. ábra. Klaszterezési eredmények a 0. korcsoport adatain

klaszterbe tartozó adatok mennyisége lényegesen csökken. Továbbá a B és D klaszterek közös határa megközelíti a vízszintest: a K-Means esetén a D klaszterbe tartozó nagy távolságú adatok a Mini Batch módszer esetén átkerülnek a B klaszterbe. A Spectral módszer esetén nagyobbak az eltérések: a B és D klasztereknek lényegesen nagyobb a területük, így az A-ba nagyon kevés útvonal esik. A C és E klaszterek határa lentebb kerül a két K-Means eredményhez képest, a nagy távolságú adatpontok a C klaszterhez fognak tartozni.

Edzés tervezés szempontjából mindenkor klaszter kialakítás felhasználható, megközelítően azonos eredményekkel. A K-Means esetén a klaszter méretek közelebb esnek egymáshoz mint a másik 2 módszerrel, így a javasolt edzést minden klaszter esetén megfelelő mennyiségű adat támásztja alá, azonban a Spectral algoritmus lényegesen jobban körbehatárolható csoportokat alakított ki.



3.5. ábra. Klaszterezési eredmények a 1. korcsoport adatain

3.3.2. 1. korcsoport

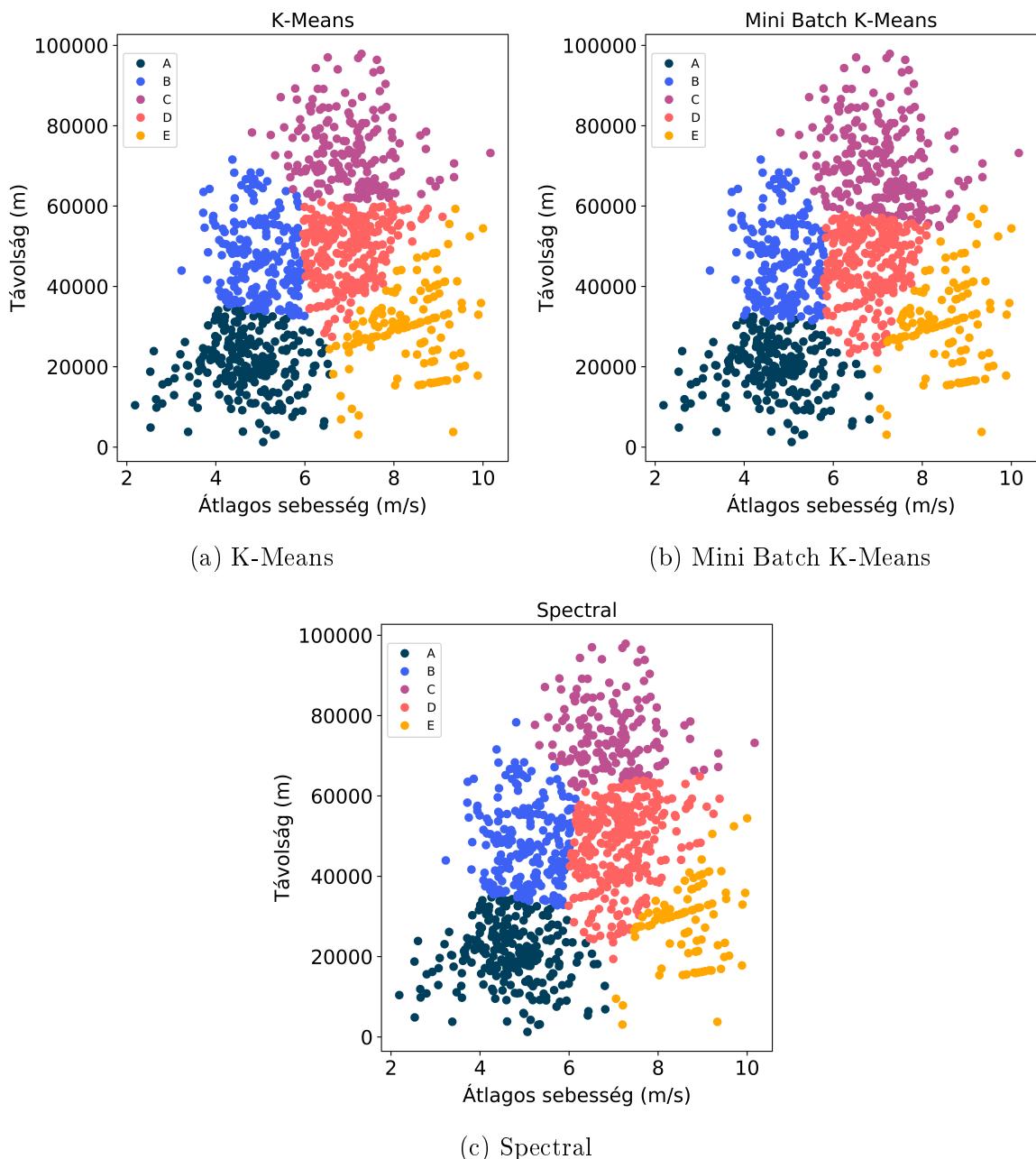
A részhalmaz összesen 7219 útvonalat tartalmaz. A különböző klaszterezési eljárások eredményei a 3.5. ábrán láthatóak. A kialakított klaszterek elhelyezkedése a két K-Means esetén nagy vonalakban megegyezik, ezeket általánosan az alábbi módon lehetne jellemzni:

- A : normális és nagy távolság - normális és extra magas átlagsebesség
- B : nagy távolság - normális átlagsebesség
- C : extrém nagy távolság - magas és extrém magas átlagsebesség
- D : normális távolság - magas átlagsebesség

- E : normális és nagy távolság - extrém magas átlagsebesség

Ezzel szemben a 3.5c. ábrán látható Spectral klaszterezési eredmények nagy mértékben különbözőek, érdekes módon az A klaszter két részre szakad. Ez annak az eredménye lehet hogy a Spectral eljárás a klaszterek kialakítása során dimenzió redukálást végez, és a csökkentett dimenziószám alapján keresi az kapcsolódó adatpontokat. Ez az eredmény edzés tervezéshez nem kifejezetten használható, a kialakult klasztereket nem lehet könnyen körbehatárolni.

3.3.3. 2. korcsoport



3.6. ábra. Klaszterezési eredmények a 2. korcsoport adatain

A részhalmaz összesen 1050 útvonalat tartalmaz. A különböző klaszterezési eljárá-

sok eredményei a 3.6. ábrán láthatóak. A kialakított klaszterek kis mértékben térnek el egymástól, a következő képen lehet jellemzni őket:

- A : normális távolság - normális átlagsebesség
- B : nagy távolság - normális átlagsebesség
- C : extrém nagy távolság - magas és extrém magas átlagsebesség
- D : nagy távolság - magas átlagsebesség
- E : normális távolság - extrém magas átlagsebesség

A K-Means és a Mini Batch módszerek eltérése főleg a D klaszter esetében figyelhető meg: a K-Means eredményeihez képest a felső határ lentebb hajlik, a nagy távolságú és magas sebességű pontok átkerülnek a C klaszterbe, valamint ezzel egyidejűleg az E klaszterből néhány kis távolságú pont a D klaszterbe kerül.

A K-Means variációk és a Spectral klaszterezés különbsége ezen az adathalmazon szintén a D klaszter kiemelésével figyelhető meg a legegyszerűbben. A D klaszter gyakorlatilag minden irányba kitágul, a szomszédos csoportok határoló pontjai átkerülnek hozzá, ez alól kivételt csak a B C D klaszterek találkozása képez.

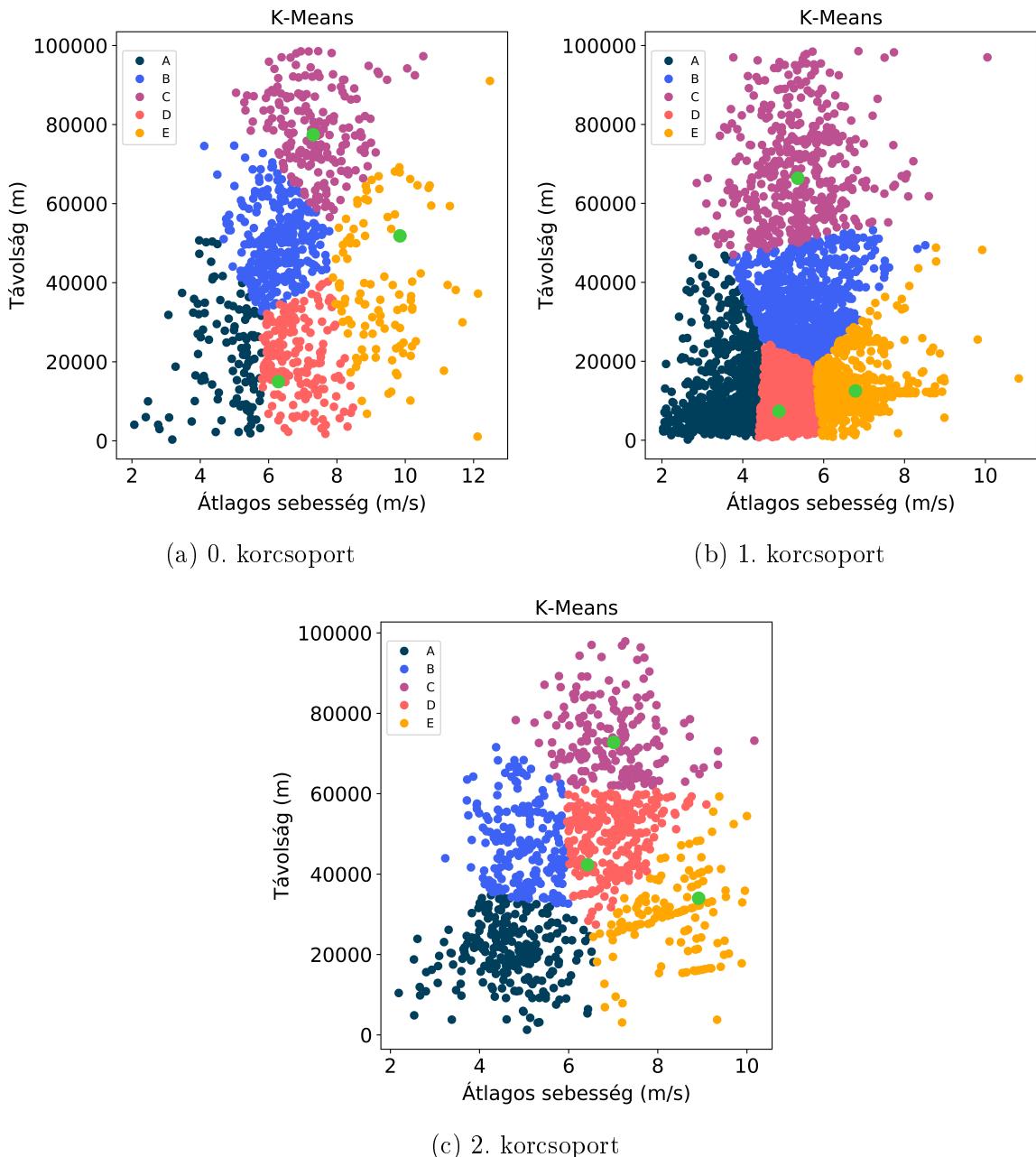
Az eredményül kapott klaszterek mindegyike felhasználható edzés javaslat készítésére, természetesen kicsit eltérő eredményekkel.

3.3.4. Összefoglalás

A három korcsoport alapján készült klaszterek eltérései jól láthatóak a 3.4., 3.5., és a 3.6. ábrán. Felhasználásukkal minden korcsoportra egyedi edzés javaslat készíthető, figyelembe véve hogy a felhasználó az egyes klaszterek jellemzése alapján milyen jellegű és nehézségű edzést szeretne végezni. Az edzés javaslatok készítésére a 3.7. program részlet szolgál, ennek néhány eredménye az alábbiakban látható. Mind a három korcsoportra 3 edzés javaslat készült az C, D és E típusú klaszterek alapján, minden esetben C - normál, D - könnyű és E - nehéz fokozattal.

A 0. korcsoportra készült edzés javaslatok a 3.7a. ábrán láthatóak. 3 javaslat készült, a C, D és E klaszterek alapján, különböző nehézségi fokozatokkal.

- C klaszter: extrém nagy távolságú, magas és extrém magas átlagsebességű útvonalak, a módszer egy normál fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 77410.3 méter (77.4 km)
 - átlagos sebesség: 7.3 m/s (26.28 km/h)
 - mozgási idő: 2 óra és 57 perc
- D klaszter: normális távolságú, magas átlagsebességű útvonalak, a módszer egy könnyű fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 14982.8 méter (14.98 km)
 - átlagos sebesség: 6.3 m/s (22.68 km/h)
 - mozgási idő: 40 perc



3.7. ábra. Edzés javaslatok a K-Means klaszterezés eredményei alapján

- E klaszter: normális és nagy távolságú, extrém magas átlagsebességű útvonalak, a módszer egy nehéz fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 51853.1 méter (51.85 km)
 - átlagos sebesség: 9.8 m/s (35.28 km/h)
 - mozgási idő: 1 óra és 28 perc

A 1. korcsoportra készült edzés javaslatok a 3.7b. ábrán láthatóak. 3 javaslat készült, a C, D és E klaszterek alapján, különböző nehézségi fokozatokkal.

- C klaszter: extrém nagy távolságú, magas és extrém magas átlagsebességű útvonalak, a módszer egy normál fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:

- távolság: 66389.3 méter (66.39 km)
 - átlagos sebesség: 5.3 m/s (19.08 km/h)
 - mozgási idő: 3 óra és 26 perc
- D klaszter: normális távolságú, magas átlagsebességű útvonalak, a módszer egy könnyű fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 7350.95 méter (7.35 km)
 - átlagos sebesség: 4.9 m/s (17.64 km/h)
 - mozgási idő: 25 perc
 - E klaszter: normális és nagy távolságú, extrém magas átlagsebességű útvonalak, a módszer egy nehéz fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 12489.7 méter (12.49 km)
 - átlagos sebesség: 6.8 m/s (24.48 km/h)
 - mozgási idő: 31 perc
 - C klaszter: extrém nagy távolságú, magas és extrém magas átlagsebességű átlagsebességű útvonalak, a módszer egy normál fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 72773.5 méter (72.77 km)
 - átlagos sebesség: 7.0 m/s (25.2 km/h)
 - mozgási idő: 2 óra és 53 perc
 - D klaszter: nagy távolságú, magas átlagsebességű útvonalak, a módszer könnyű fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 42353.25 méter (42.35 km)
 - átlagos sebesség: 6.4 m/s (23.0 km/h)
 - mozgási idő: 1 óra és 50 perc
 - E klaszter: normális távolság, extrém magas átlagsebességű útvonalak, a módszer egy nehéz fokozatú edzésre tett javaslatot. A javasolt edzés paraméterei:
 - távolság: 33997.5 méter (34.0 km)
 - átlagos sebesség: 8.9 m/s (32.04 km/h)
 - mozgási idő: 1 óra és 4 perc

A kapott edzés javaslatok között jól látható különbség van, az egyes korcsoportokra jellemző útvonalakhoz igazodva egyedi eredményeket adnak. Az edzés tervezés már ezzel az alapvető módszerrel megvalósítva is a felhasználó kora alapján skálázódik, ezzel korcsoportokra szabott, specifikus javaslat tételt téve lehetővé.

4. fejezet

Összefoglalás

A szakdolgozat során alkalmazott gépi tanulási módszerek várható módon eltérő eredményekkel szolgáltak azonban szinte mindegyikre nézve volt olyan terület ahol kiemelkedően jobban teljesítettek mint az adott problémára kipróbtált egyéb eljárások. Azonban a gépi tanulás alkalmazása során nincs kőbe vésve hogy egy adott problémára mi a legjobb megoldás, csupán kiindulási pontot lehet meghatározni a területhez kapcsolódó iránymutatásra és gyakorlatra támaszkodva, végső soron pedig az eredmény és a kiválasztott eljárás a felhasznált adathalmaztól függ.

A szakdolgozat eredményei jól mutatják hogy a gépi tanulás kiválóan használható nem csak a versenyszerű hanem a minden nap sport esetében is, alkalmazásával kialakíthatóak olyan funkciók amelyek segítenek az átlagos, hobbi szerűen kerékpározóknak az útjaik megtervezésében, edzéseik kialakításában. Ehhez természetesen alapfeltétel egy nagy mennyiségű, változatos adathalmaz megléte, azonban a kerékpározók körében annyira természetes az útvonalai rögzítése hogy ez nem jelenthet akadályt.

A kerékpározáshoz kapcsolódó alapvetőbb számítások mint a különböző idő intervallumok meghatározása egyszerű regressziókkal könnyen elvégezhetők és a viszonylag kis méretű adathalmazon is nagy pontosságot eredményeztek. Az összetettebb feladatok megoldása nem ennyire kézenfekvő, a különböző típusú nehézségi osztályok meghatározására nincs egyértelmű jó megoldás, azonban a szakdolgozat során született eredmények egy használható alapot adnak, melyek később továbbfejlesztésre kerülnek. A módszerek és eredmények fejlesztését magába foglaló projekt túlmutat a szakdolgozat keretein, azonban magába foglalná egy web vagy mobil alapú alkalmazás elkészítését, amely lehető tenné a felhasználói számára a szakdolgozat során kidolgozott megoldások használatát. Egy ilyen alkalmazás a felhasználók kiszolgálásán kívül a megoldások fejlesztésének megkönnyítését is szolgálná, mivel egy növekvő felhasználói bázis könnyen sokszorosára emelhetné a a gépi tanulás során felhasználható adatmennyiséget, ezzel javítva az adathalmaz sokszínűségét, lehetővé téve pontosabb nehézségi csoportok kialakítását nem és korcsoportok szerint. A nehézségi csoportok fejlődése hosszútávon hozzájárulna egy összetettebb edzési javaslatokat készítő rendszer megvalósításához, amely a távolságon és átlagos sebességen felül az út egyéb paramétereire is megbízható ajánlást tudna készíteni, akár konkrét, lokáció alapú útvonalrajzolással.

Irodalomjegyzék

- [1] Bike Calculator webes és mobilos alkalmazás
<http://bikecalculator.com>
- [2] Strava sporttevékenység követő alkalmazás
<https://www.strava.com>
- [3] Endomondo sporttevékenység követő alkalmazás
<https://www.endomondo.com/>
- [4] Best Bike Split versenyre készülést segítő alkalmazás
<https://www.bestbikesplit.com>
- [5] Scikit-learn: Python gépi tanulás könyvtár
(Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.)
<https://scikit-learn.org>
- [6] Scikit-learn Ridge regresszió
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
- [7] Ridge regresszió
<https://www.statisticshowto.datasciencecentral.com/ridge-regression>
- [8] Scikit-learn Lasso regresszió
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html
- [9] Lasso regresszió
<https://www.statisticshowto.datasciencecentral.com/lasso-regression/>
- [10] Scikit-learn Random Forest regresszió
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
- [11] Random Forest regresszió
https://turi.com/learn/userguide/supervised-learning/random_forest_regression.html
- [12] Scikit-learn klaszterezés útmutató
<https://scikit-learn.org/stable/modules/clustering.html>
- [13] Scikit-learn KMeans klaszterezés
<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

-
- [14] Scikit-learn Mini-Batch K-Means klaszterezés
<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans>
 - [15] Scikit-learn Spectral klaszterezés
<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering>
 - [16] Spectral klaszterezés
<https://towardsdatascience.com/spectral-clustering-82d3cff3d3b7>
 - [17] Spectral és K-Means klaszterezés összehasonlítása
<http://scalefreegan.github.io/Teaching/DataIntegration/practicals/p2.html>
 - [18] JQuery JavaScript könyvtár
<https://jquery.com/>
 - [19] Google Firebase
<https://firebase.google.com/>
 - [20] Pandas: Python adat analízis könyvtár
<https://pandas.pydata.org/>
 - [21] Matplotlib: Python grafikus ábrázolás könyvtár
<https://matplotlib.org/>
 - [22] Angol kifejezések magyarítása
<https://www.tankonyvtar.hu/hu/tartalom/tkt/angol-magyar/ch19.html>
 - [23] Random Forest regresszió paraméter hangolás
<https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/>

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