Pannon Egyetem
Műszaki Informatikai Kar
Rendszer- és Számítástudományi Tanszék
Programtervező informatikus MSc

Diplomadolgozat

Futóverseny eredményeinek kiértékelését támogató eszköztár és alkalmazás fejlesztése

Csizmazia Máté

Témavezető: Dr. Leitold Dániel

2022

PANNON EGYETEM
MŰSZAKI INFORMATIKAI KAR
Programtervező informatikus MSc szak

Veszprém, 2022. március 23.

DIPLOMADOLGOZAT TÉMAKIÍRÁS

Csizmazia Máté
Programtervező informatikus MSc szakos hallgató részére

Futóverseny eredményeinek kiértékelését támogató eszköztár és alkalmazás fejlesztése

Témavezető: Dr. Leitold Dániel, adjunktus

A feladat leírása:

A hallgató feladat futóverseny adatok begyűjtése, tisztítása, rendezése, adatelemzésre előkészítése. Ismerje meg a felügyelt tanulási módszereket, és válasszon ki egy alkalmas módszertant az alábbi feladatnak megfelelően: a meglévő adatok alapján olyan modell építése, amely képes új versenyzők adatai alapján előre jelezni, hogy az aktuálisan megadott (új) versenyző képes lesz-e az adott versenyt (amire a modell épül) teljesíteni, lefutni. Értékelje ki a modell pontosságát, amennyiben lehetőség nyílik alkalmazzon több módszertant, vesse össze a kapott eredményeket, illetve amennyiben szükséges, fejlessze tovább a meglévő módszereket az eredmények javítása érdekében.

Feladatkiírás:

- Dolgozza fel a témával kapcsolatos eddigi hazai és külföldi irodalmat!
- Határozza meg, gyűjtse be, tisztítsa meg, készítse elő az adathalmazt!
- Tervezze meg az adatelemzés folyamatát!
- Építsen osztályozó modellt, modelleket a meglévő adatok alapján!
- Értékelje ki a létrehozott osztályozó modellt, modelleket!
- Tegyen javaslatot a meglévő modellek továbbfejlesztésére!

Dr. Hartung Ferenc
egyetemi tanár
szakfelelős

Dr. Leitold Dániel
adjunktus
témavezető

Hallgatói nyilatkozat

Alulírott *Csizmazia Máté* hallgató kijelentem, hogy a dolgozatot a Pannon Egyetem *Rendszer- és Számítástudományi* tanszékén készítettem az *okleveles programtervezö informatikus* végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Dátum: Veszprém, 2022. május 10.

.....

Csizmazia Máté

Témavezetői nyilatkozat

Alulírott *Dr. Leitold Dániel* témavezető kijelentem, hogy a dolgozatot *Csizmazia Máté* a Pannon Egyetem *Rendszer- és Számítástudomány* tanszékén készítette az *okleveles programtervezö informatikus* végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Dátum: Veszprém, 2022. május 10.

.....

Dr. Leitold Dániel

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani mindazoknak, akik szakmailag és erkölcsileg hozzájárultak a diplomadolgozatom megvalósításához. Kitartó munkájuknak, segítőkészségüknek, és szellemi támogatásuknak köszönhetően jöhetett létre a dolgozat.

Köszönöm témavezetőmnek, Dr. Leitold Dánielnek a sok segítséget, támogatást és építő kritikát. Munkája és szakmai hozzáértése nélkül nem születhetett volna meg ez a dolgozat és a hozzá tartozó szoftver.

Továbbá hálával tartozom családomnak a kitartó szellemi támogatásért és azért, hogy lehetővé tették tanulmányaimban való előre haladásom.

Tartalmi összefoglaló

A futás az emberek által egyik legszélesebb körben űzött edzésfajtája. Nem véletlen, hiszen mindenki számára könnyedén elérhető, nincs szükség speciális felszerelésre és meglehetősen jó hatással van az egészségre is. Magyarországon létezik egy népszerű hosszútávú futóverseny, ami a BSZM (másnéven Balaton Szupermaraton) névre hallgat. Ez az ultramarathon egy igazán érdekes kihívás minden hosszútávú futó számára. Kihívás, hiszen közel 200 kilométernyi táv lefutását jelenti, 4 nap lefolyása alatt. Emiatt találtam fontosnak, hogy elkészüljön a futóverseny eredményeinek kiértékelését támogató eszköztár és alkalmazás.

A dolgozat célja, hogy az ilyen futómaratonok eredményeit részletesebben feltárja és jól vizualizálhatóvá alakítsa. A dolgozathoz egy olyan komplex felhasználóbarát alkalmazás fejlesztése a cél, amely lehetővé teszi az eredmények kiértékelését és historikus adatok alapján prediktív előrejelzéseket tud adni a felhasználó számára. A predikciót különböző, gépi tanuló algoritmusok segítségével tudjuk megvalósítani, így a végfelhasználó számára többnyire pontos visszajelzést tudunk adni, a jövőbeli verseny kimeneteléről. Az előrejelzés pontosításának az érdekében különböző mintavételezési és keresztvalidációs eljárások alkalmazására volt szükség.

Az alkalmazás elkészítéséhez a manapság leginkább elterjedt és legnépszerűbb programozási nyelvet fogom igénybe venni, ami nem más, mint a Python. A választásom azért esett erre a technológiára, mivel népszerűségének köszönhetően hatalmas felhasználó bázissal rendelkezik, egyszerű a használata, és ez a nyelv a leginkább támogatott az adatbányászat és Data Science területén. Ezek mellett, rengeteg keretrendszer és könyvtár áll rendelkezésre, ami segíti a fejlesztés menetét.

Kulcsszavak: [BSZM, Gépi tanulás, Data Science, Osztályozás, Python]

Abstract

Running is one of the most popular types of excercise practiced by people. It is no surprise, since it is easily accessible to everyone, does not require special equipment and has a fairly good effect on health as well. There exists a popular long-distance running competition in Hungary, namely BSZM (also known as Balaton Supermarathon). This ultramarathon is a real challenge for any long distance runner. It is difficult, as it means covering a distance of almost 200 kilometers in 4 days. For this reason, a toolkit and and application to support the evaluation of the results of the competition was created. The aim of the dissertation is to explore the results of such running marathons in more detail and to make them well visualizable.

The goal is to develop a complex user-friendly application that allows the evaluation of the results and can provide predictive predictions to the user based on historical data. Predictions can be implemented using various machine learning algorithms, so we can give the end user accurate feedback on the outcome of future competitions. Different sampling and cross-validation procedures were required to refine the prediction accuracy.

To create the application, I will use one the most common and popular programming language today, which is none other than Python. I chose this technology because of its popularity, it has a huge user base and community, it is easy to use, and it is the most supported language in Data Mining and Data Science. In addition, there are plenty of frameworks and libraries available to help with the development process.

Keywords: [BSZM, Machine Learning, Data Science, Classification, Python]

Tartalomjegyzék

1. Bevezetés ... 10

2. Data Science ... 12

2.1. Adatbányászat ... 13

2.1.1. Főbb adatbányászati feladatok ... 13

2.2. Web Scraping ... 15

3. Futóversenyek és elemzéseik ... 16

3.1. Maraton elemzések ... 16

3.2. Ultramaraton elemzések ... 16

3.3. Balaton Szupermaraton ... 17

4. Technológiák ... 18

4.1. Python ... 18

4.2. Jupyter Notebook ... 18

4.3. Visual Studio Code ... 19

4.4. Pandas ... 19

4.5. Plotting könyvtárak ... 20

4.6. DataPrep ... 20

4.7. Scikit-learn ... 20

4.8. Streamlit ... 20

5. Célkitűzés ... 22

6. Megvalósítás ... 24

6.1. Első iteráció ... 24

6.1.1. Adathalmaz ... 24

6.1.2. Adatmigráció ... 25

6.1.3. Adattisztítás és transzformáció ... 25

6.1.4. Adatdiszkretizáció ... 26

6.1.5. Feltáró elemzés ... 26

6.1.6. Osztályozás ... 27

6.1.7. Kiértékelés ... 29

6.2. Probléma ... 29

6.2.1. Kis adathalmaz 29
6.2.2. Kiegyensúlyozatlan adathalmaz 30
6.2.3. Zajos adathalmaz 30
6.3. Második iteráció 30
6.3.1. Adathalmaz bővítése 31
6.3.2. Adattisztítás és transzformáció 32
6.3.3. Outlier analízis 33
6.3.4. Feature engineering 33
6.3.5. Feltáró elemzés 33
6.3.6. Importance analysis 34
6.3.7. Tulajdonság kiválasztás 35
6.3.8. Keresztvalidálási eljárások 35
6.3.9. Osztályozás 38
6.3.10. Kiértékelés 40

7. Alkalmazás 41

7.1. Telepítés 41
7.2. Konfiguráció 41
7.3. Streamlit 42
7.4. Futtatás 42
7.5. Felhő 42
7.6. Felületek 43
7.6.1. Statisztika 43
7.6.2. Modellek 44
7.6.3. Előrejelzés 45
7.6.4. Adathalmaz feltöltés 45
7.6.5. Információk 46

8. Összesítés 47

9. Továbbfejlesztési lehetőségek 49

9.1. Hiperparaméter hangolás 49
9.2. Új célok megfogalmazása 49
9.3. További együttműködés a szervezőkkel 50
9.4. Alkalmazás továbbfejlesztése 50

Irodalomjegyzék..... 52

Mellékletek 54

Ábrajegyzék..... 55

Táblázatjegyzék 56

1. Bevezetés

A maraton talán a leghíresebb versenyfajtája a futásnak. Eredete egészen az ókori Görögországig visszavezethető. A hírnevét a futószám távja miatt érdemelte ki magának, mivel egy ilyen hosszúságú táv teljesítése többek szerint közel emberfeletti teljesítményt szimbolizál, emiatt is vált ez a sportág a kitartás egyik fő szimbólumává.

Létezik a maratonnak egy ennél is megerőltetőbb verziója, ami nem más, mint az ultramarathon. Erről azt érdemes tudni, hogy egy ilyen esemény általában egy többnapos versenyszám, amit lehet csoportosan váltóban is végezni, de a legedzettebbek egyénileg is képesek teljesíteni. A szupermaraton alatt megtett kilométer összesen meghaladhatja a 200-at is néhány esetben, ami egy 4 napos rendezvényen napi legalább egy egész maratonnyi távot jelent. Érezhető, hogy egy ilyen feladat teljesítése talán még a legedzettebb futóknak is nagy kihívást jelenthet.

A dolgozat kiindulásához egy olyan adathalmaz áll rendelkezésre, ami egy ilyen szupermaraton eredményeit tartalmazza. Ez a futóverseny a BSZM (Balatoni Szupermaraton), ami lényegében a Balaton körbefutásáról szól. Ez egy négynapos esemény és távban körülbelül 180-200 kilométert jelent. A táv kissé eltérhet a különböző évek adatai között, mivel az útvonalat az évek során több esetben is megváltoztatták, ami a táv növekedését vagy csökkenését eredményezte.

A dolgozat elkészítésének az a fő célja, hogy ezen adathalmazok feldolgozásával nagyobb rálátást nyújtson az eredményekre, jól vizualizálhatóvá alakítsa ezt és hasznos vagy érdekes új információkat tárjon fel. Ehhez, egy olyan komplex alkalmazás elkészítése társul, ami lehetővé teszi az eredmények felhasználó által állítható kiértékelését, és historikus adatok alapján prediktív előrejelzéseket is tud adni a felhasználó számára.

Ahhoz, hogy egy ilyen alkalmazást el lehessen készíteni, a manapság egyre népszerűbb és széles körben alkalmazott adatbányászati eljárásokat fogjuk felhasználni. Ez a technológia jelenlegi fénykorát éli az informatika világában, emiatt számos nagyvállalat számára is jó stratégiának számít. Alkalmazható kereskedelem, orvostudomány, autóipar, sport vagy számos más területeken is egyaránt.

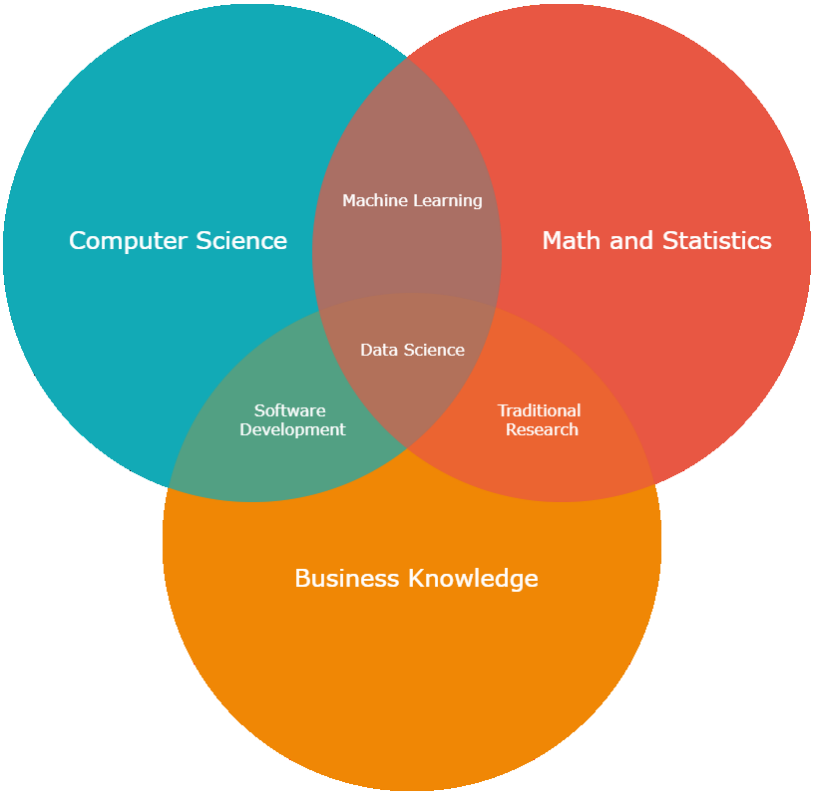
Az alkalmazás elkészítéséhez manapság az egyik legnépszerűbb programozási nyelvet fogom használni, ami nem más, mint a Python. Egyértelmű volt, hogy ezzel a technológiával valósítsam meg az alkalmazást, mivel előzetesen már volt lehetőségem

alkalmazni egyéni projektjeim során és hatalmas felhasználói bázissal rendelkezik, ami nagy segítséget tud jelenteni a fejlesztésben. Ezek mellett a használata és szintaxisa a nyelvnek meglehetősen egyszerű. A Data Science és az adatbányászat területén is kiemelkedő támogatásnak örvend a Python, de természetesen nem ez az egyetlen programozási nyelv, amin alkalmazni lehet.

Számos könyvtár áll rendelkezésre a Python nyelvhez, amik megkönnyítik a munkát. Ilyen például a pandas ami főképp olyan feladatokat lát el, aminek a segítségével könnyedén tudjuk az adathalmazt manipulálni, transzformálni és tisztítani. Másik példa a matplotlib, ami kifejezetten abból a célból jött létre, hogy az adatvizualizációt segítse rengeteg féle diagram segítségével. A scikit-learn egy másik gyakran használt könyvtár, aminek a segítségével gépi tanuló modelleket lehet konstruálni. Ezeken felül természetesen számos más hasznos könyvtár is létezik, amikről a későbbi fejezetekben lehet olvasni.

2. Data Science

A Data Science egy olyan tudományos ágazat, ami kifejezetten a nagy mennyiségű adatok feldolgozásával és értelmezésével foglalkozik. Ez a rövid megfogalmazása, de a valóságban sok apróbb témát is lefed a kifejezés. Ha pontosabban szeretnénk vizualizálni, hogy mégis milyen fő részekből épül össze ez a tudomány, ezt legegyszerűbben egy Venn diagram segítségével tudjuk megtenni. [1]



1. ábra Data Science Venn diagram

Számítástechnika

Ez talán a legtriviálisabb alkotóeleme ennek a tudománynak. Nélkülözhetetlen az alkalmazása, mivel ekkora adathalmazok feldolgozását csak számítógépek segítségével lehet optimálisan és könnyedén végrehajtani.

Matematika és statisztika

A matematika adja meg az alapját a Data Science-nek. A matematikának 3 fő részét érdemes kiemelni, ami a kalkulus, a lineáris algebra és a valószínűségszámítás / statisztika. Azért elengedhetetlen a használata, mivel a matematikának léteznek olyan eszközei, amivel kiszámolhatjuk a nagy adatok közötti összefüggéseket, mintákat találhassunk az adatok között, vagy következtetéseket vonjunk le a feldolgozott adatok

alapján. Olyan fontos matematikai fogalmak fognak előkerülni a későbbi fejezetekben, mint az átlag, medián, szórás, korreláció analízis, valószínűségszámítás és még rengeteg más is.

Üzleti tudás

Ahhoz, hogy a lehető legeffektívebben lehessen alkalmazni a technológiát, elhanyagolhatatlan a területi vagy üzleti ismereteknek a megléte.

Az ábrán látható metszetek többnyire maguktól értetődőek, hiszen szoftverfejlesztéshez az informatikai és a területi ismeretek elegendőek ahhoz, hogy egy üzleti alkalmazás elkészüljön, gépi tanuló algoritmusok elkészítéséhez matematikára és informatikára van szükség, és nem utolsó sorban, tradicionális módon is lehet kutatást végezni, de ez nem a legoptimálisabb eljárás manapság. A három főtémakör metszete pedig nem más, mint maga a Data Science.

2.1. Adatbányászat

Az adatbányászati algoritmusokat az adathalmaz vagy adatbázisból való tudásfeltárás során alkalmazzák. Ez egy olyan folyamat, amelynek során valamilyen újszerű, hasznos információt vagy mintát sikerül felfedezni. Napjainkban leginkább hasznosított területe az üzleti szférában érzékelhető, de előszeretettel használják minden területen, legyen szó genetikáról, orvostudományról vagy sportról.

2.1.1. Főbb adatbányászati feladatok

Az adatbányászat egy több lépésből álló folyamat. Ahogy már említettem, nagy adathalmazokat feldolgozó feladatokat végez. Ekkora mennyiségű adat összegyűjtése, nem minden esetben áll rendelkezésre, ezért sokszor a munka az adatok összegyűjtésével kezdőik.

Adatelőkészítés

Amennyiben megvan az adathalmaz, ezt a további lépésekben elő kell készíteni mielőtt adatbányászati algoritmusokat alkalmaznánk rajtuk, mivel ezek csak az adatokban ténylegesen jelenlévő mintákat képesek feltárni. Az előfeldolgozás (pre-processing) elengedhetetlen lépése a folyamatnak, ahol az adatkészletet megtisztítják és transzformálják.

Gyakori minták kinyerése

Cél megtalálni gyakran előforduló objektumokat. Ezt a legegyszerűbben úgy lehetséges, hogy vizualizáljuk az adatokat diagramok segítségével.

Attribútumok közötti összefüggések

Attribútumok, másnéven tulajdonságok között azért érdemes megvizsgálni a kapcsolatokat, mivel ezáltal lehetőségünk van pótolni a hiányos adatokat és lehet javítani a zajokat. Többféle típusa is létezik a folyamatnak, általában az adattípusnak megfelelő eljárást érdemes használni. Skála típusú változók esetében az összefüggést korreláció és regresszió analízissel tudjuk kimutatni. Ordinális típusú változóknál rangkorrelációs mutatók mérik ezt.

Csoportosítás

Csoportosítás folyamán az adatokat külön klaszterekbe kell elválasztanunk úgy, hogy az egy csoportba tartozó elemek mind hasonlóak legyenek, de a különböző csoportokban lévők pedig különbözzenek egymástól.

Osztályozás

Osztályozás eljárás során arra törekszünk, hogy egy rekordot, előzetes adatok alapján, valamilyen osztályba tudjunk sorolni. Erről a későbbi fejezetekben bővebben lehet olvasni. [1]

Regresszió

A regresszióanalízis segítségével egy függő változó és vagy több független változó közötti kapcsolatok becslése végezhető. Több változata is létezik, de a legelterjedtebb formája a lineáris regresszió, amelynek során egy olyan egyenes megtalálása a cél, amely a legjobban illeszthető az adatokhoz.

Dimenziócsökkentés

A dimenziócsökkentés az adathalmaz olyan átalakítását jelenti, amely során a nagy dimenziójú térből egy alacsony dimenziós teret hozunk létre úgy, hogy az alacsony dimenziós ábrázolás megtartja az eredeti adatok fontos tulajdonságait, ideálisan a belső dimenziójához közel.

Szövegbányászat

A szövegbányászat manapság egyre inkább elterjedt módszer. Lényege, hogy a strukturáltalan szöveget strukturált formátumúvá alakítsa, hogy értelmes mintákat tudjon felfedezni bennük. Számos algoritmus segíti a vállalatokat, mint a Naive Bayes vagy a

Support Vector Machines, hogy felfedezhessenek strukturált adataikon belüli rejtett kapcsolatokat.

2.2. Web Scraping

Amint az előző fejezetben említettem, az adatok beszerzése sokszor bonyolultabb lehet, mint gondolnánk. Amennyiben szerencsénk van, ezt készen is elérhetjük csv formátumban, de sajnos a legtöbb esetben ez nem így van. Ilyenkor segítségünkre vannak olyan technológiák, amelyek segítségével automatizált módon összegyűjthető az adat, legyen szó bármilyen weboldalról. Ez a technológia nem más, mint a webscraping. Ahogy a neve is utal rá, itt ténylegesen a weboldal forráskódjából „kaparjuk” le az adatokat, valamilyen automatizált algoritmus segítségével.

Selenium

A selenium talál a legelterjedtebb könyvtár, amivel lehet webscraping-et alkalmazni. Több programozási nyelvet is támogat, mint a Java, Python, C# stb. Ez egy ingyenes nyílt forráskódú keretrendszer, amit több céllal is lehet használni, de a mi esetünkben, csak a webscraping-re lesz alkalmazva. Működése meglehetősen egyszerű, mivel kódon belül, egyszerűen manipulálható teljes mértékben a működése. Néhány böngésző esetében engedélyezni kell, hogy automatikusan tudjon műveleteket végrehajtani, erre példa a Safari, de ez is pár kattintással végrehajtható. Többnyire az összes böngészőt támogatja, tehát bárki számára elérhető.

A seleniumot alkalmazva gyakorlatilag bármilyen weboldalról tudunk automatizálva adatokat begyűjteni, ahol táblázatosan kilistázható az információ. Innentől csak annyi dolgunk van, hogy megírjuk az algoritmust, ami a HTML-ből kinyeri az adatokat, egy közös szerkezetre hozza, és végül kiexportálja egy olyan formátumba, amit már fel tudunk dolgozni egyenesen pythonnal.

3. Futóversenyek és elemzéseik

A futóversenyek népszerűségének köszönhetően rengeteg adatot lehet találni. Olyan futóversenyek eredményei is elérhetőek, mint például a Bostoni maraton, ahol éves szinten több tízezren futnak együtt, hogy teljesítsék ezt a versenyszámot. A nagy adatmennyiségnek köszönhetően, természetesen többen is neki láttak, hogy különböző kimutatásokat, elemzéseket készítsenek. Az elemzésekhez használt legnépszerűbb közösségi felület a kaggle, ezért én is itt kezdtem hasonló kutatások és statisztikai kimutatások után keresni. [2]

3.1. Maraton elemzések

A kutatás során rábukkantam különféle munkákra. Kifejezetten érdekesnek találtam azt a munkát, amely kifejezetten arra törekedett, hogy egy olyan előrejelző mesterséges intelligencia, amely egy futó jövőbeli maraton eredményét legyen képes megjósolni a lehető legpontosabban. Ehhez a készítő egy 45 másodperces hibahatárt fogalmazott meg, és igyekezett a modellt különféle technikákkal addig pontosítani, amíg ezt nem tudta teljesíteni. Ezt többnyire sikerült is betartania. Egy másik érdekes kimutatásban a Bostoni maratoni eredményeket tették jól vizualizálhatóvá különböző diagramok felhasználásával. Olyan kutatás is létezik, amely a legjobban teljesítő korcsoportot és kort kereste a maratoni futószámokon és ennek a kidolgozását vezette végig. [2]

3.2. Ultramaraton elemzések

Ultramarathon eredményeit már sokkal ritkábban elemezték a felhasználók, emiatt meglehetősen nehézkes volt bármiféle hasonló kutatási eredmény elsajátítása. Lehetett találni egyszerű kimutatásokat az ilyen versenyszámok esetén is, de ezek nagyban eltérnek az afféle osztályozási feladatoktól, mint amivel a dolgozat foglalkozik. Ebből kifolyólag a dolgozat témája egyedinek tekinthető, mivel az irodalom még nem dolgozott fel, ilyen módon ultramarathon eredményeket. [3]

3.3. Balaton Szupermaraton

A balatoni ultramarathon egy kifejezetten népszerü kihívás a futók körében, viszont ehhez köthetö elemzés, vagy kutatás eddig nem látott napvilágot. A futóverseny eredményei publikusan elérhetök mindenki számára. Azért esett a választásom kifejezetten az egyéni eredményeket tartalmazó adathalmazra, mivel egy érdeklődő személy kifejezetten ennek a versenyszámnak a részleteire volt kíváncsi. Egy olyan részletes elemzésre volt szüksége, amivel tisztán láthatóvá válhat, hogy a jelenlegi teljesítménye elegendő-e egy ilyen nagy kihívás teljesítésére. Emellett érdekelték olyan érdekességek is mint, hogy milyen eloszlásúak a futók sebességei vagy milyen arányban teljesítették a hasonló korabeli futók a versenyt. Ezen felül számos érdekesnek tekinthető információ is kinyerhető ebből a kisebbnek nevezhető adathalmazból.

4. Technológiák

Ahhoz, hogy a már meglévő adathalmaz feldolgozásra kerülhessen, alkalmaznunk kell különböző technológiákat, könyvtárakat és keretrendszereket. Mivel a python a leginkább támogatott nyelv a Data Science területén, ezért egyértelmű volt, hogy ezt alkalmazzam a fejlesztéshez.

4.1. Python

A python manapság a világ egyik legismertebb és legtöbbet használt programozási nyelve. Nem véletlen, hiszen alkalmazása és szintaxisa meglehetősen egyszerű, így vonzó lehet a tapasztalatlan, kezdő fejlesztőknek. Természetesen nem csak kezdő szintű fejlesztők alkalmazzák, mivel így is egy egyszerű, de magas szintű, nyílt forráskódú nyelvről beszélünk.

A python már számos területen bizonyította hasznosságát:

- Webfejlesztés,
- Játékfejlesztés,
- Tudományos számításokat végző alkalmazások,
- Mesterséges intelligenciához köthető tudományok,
- Asztali alkalmazások
- Nagyvállalati alkalmazások
- Operációs rendszerek
- Szkriptírás

Amint említettem, ez a nyelv rendelkezik a legtöbb olyan könyvtárral és keretrendszerrel, ami megkönnyíti az adatbányászati feladatokat.

4.2. Jupyter Notebook

A Jupyter Notebook egy olyan nyílt forráskódú interaktív webalkalmazás, ami az adatbányászok számára lehetővé teszi olyan dokumentumok létrehozását, ami könnyedén megosztható és python kódot tud futtatni. Ez kiegészül egy markdown rendszerrel, aminek a segítségével könnyedén és jól láthatóan dokumentálható integráltan az egész munkamenet. Ennek a segítségével, könnyedén megjeleníthetővé válnak egy közös környezetben egyenletek, számítási eredmények, vizualizációk vagy képek.

Alkalmazása az összes adatbányászati feladatban hasznos, beleértve az adattisztítást, adattranszformációt, feltáró adatelemzést, adatvizualizációt, modellezést, gépi tanulást és még sok mást.

Tehát összefoglalva, egy olyan adatbányászati környezetet biztosít a felhasználó számára, ami nem csak egy fejlesztői környezetként működik, hanem egyben egy bemutatót, prezentációt segítő eszköz is.

A Notebook 2 fő összetevőből áll össze, kódcellákból és markdown cellákból. A kódcellákba írt programot a weboldal továbbítja egy háttérkernelnek, amely lefuttatja a kódot és visszaadja az eredményeket a weboldal számára. Ezeknek a kerneleknek nem feltétlenül szükséges a használt eszközön futniuk, erre tökéletes példa a Google saját hasonló működést biztosító eszköztára, ami a Google Colaboratory projektnévre hallgat. Amennyiben saját eszközről kívánja a felhasználó futtatni a Jupytert, ezt megteheti akár hálózati hozzáférés nélkül is, és helyileg tudja végezni a munkát.

4.3. Visual Studio Code

A Visual Studio Code volt az a szövegszerkesztö amire a választásom esett, mivel nyílt forráskodúságának köszönhetöen, rengeteg hasznos pluginal bövíthetö és használható. Ilyen például a Jupyter kiegészítöje is, ami technikailag egy jupyter notebook környezetet imitál, csak a szövegszerkesztőn belül. Azért preferálom jupyter VSCode-on belüli használatát, mivel így sokat segítenek a szövegszerkesztő hasznos funckiói is, mint például a kódkiegészítő, vagy a kódsnippetek billentyűparancsra való beillesztése

4.4. Pandas

A pandas a legelterjedtebb szoftverkönyvtár, amit a python nyelven belül adatmanipulációra és analízisre használnak. Számos különféle forrásból képes beolvasni adatokat, legyen szó csv-ről, xlsx-ről, vagy akár html-ről. A beolvasott adatokat egy saját DataFrame elnevézsre hallgató objektumban tárolja el, amin már a beépített műveleteket el lehet végezni. Ilyen műveletek például a több adatforrás összefűzése, adatmanipuláció, logikai vizsgálatok, adattisztítás vagy adatdiszkretizáció. Könyvtár rendkívül jól optimalizált teljesítmény szempontjából, ami miatt rendkívül jól alkalmazható nagy adathalmazok feldolgozására és módosítására.

4.5. Plotting könyvtárak

Matplotlib

Alkalmazásával gyorsan és könnyedén lehet diagramokon megjeleníteni a DataFrame-ben tárolt adatokat. Rengeteg diagram áll rendelkezésre, legyen akár szó histogramokról, kör vagy vonaldiagramokról.

Seaborn

Hasonlóan a matplotlib-hez ez is egy diagramokat készítő és megjelenítő könyvtár. Főképp csak kinézetben különbözik a másik könyvtártól.

4.6. DataPrep

Egy olyan könyvtár, ami a feltáró elemzés lépéseit teszi egyszübbé. Automatikusan készít kimutatást egy adathalmazról. Kifejezetten Jupyter notebook felhasználóknak készült, és ennek a használatára van a legjobban optimalizálva. A generált kimutatás beágyazható egyenes a notebook fájlba, vagy kiexportálható PDF, vagy egy weboldal formátumába is.

4.7. Scikit-learn

A scikit-learn egy ingyenesen elérhető gépi tanulást támogató könyvtár, ami a python programozási nyelvhez íródott. Többféle eljárást is támogat, mint osztályozás, regresszió, csoportosítás. Ezen eljárásoknak többféle fajtáját is támogatja.

4.8. Streamlit

A streamlit névre hallgató keretrendszer nemrégiben kezdett egyre inkább nagyobb népszerűségre szert tenni. A könyvtár segítségével gyorsan és könnyedén lehet adatbányászati projekteket egy web alkalmazás formájára átalakítani. Számos beépített komponensének köszönhetően szinte bármilyen feladatot látványosan meg lehet jeleníteni. Szinte minden olyan könyvtárat támogat, amit adatbányászati feladatokhoz a leginkább alkalmaznak a felhasználók.

Kifejezetten egyszerű a használata és a telepítése. A python könyvtár kezelőjének segítségével (pip) telepítés után, pár parancs kiadásával bele is lehet tekinteni milyen szolgáltatásokat nyújt a streamlit. Interaktív módon, akár egyből a felületen módosíthatóak a bemeneti értékek, szűrhetőek az adatok, vagy újra futtathatóak a

parancsok. Az alkalmazás futtatása után valójában egy webszervert indít a számítógépen, amit innentől kezdve bármilyen lokális webböngésző segítségével el lehet érni. Nagy előny, hogy amikor a script file módosításra kerül, ilyenkor a rendszer automatikusan betölti a változtatásokat az oldalon, így nem kell folyamatosan újraindítani a webszervert vagy külön bővítményeket telepíteni, amik megteszik ezt helyettünk.

Ezek mellett a streamlit szolgáltat egy külön felhő szolgáltatást is a regisztrált felhasználóknak, ahová pillanatok alatt telepíthetjük az alkalmazást. Természetesen ez csak egy bizonyos fokig ingyenes, de így is három projekt publikálására ad lehetőséget költségek nélkül. Ahhoz, hogy az alkalmazás futhasson a felhő rendszerben, nincs másra szükség, mint hogy a publikálandó projektet egy GitHub repository-ban tároljuk el. Ennek a repository-nak publikusnak kell lennie ahhoz, hogy a streamlit felhő rendszere is el tudja érni. Miután megadtuk a repository elérését a streamlit felhő kezelő oldalán, a rendszer elkezdi telepíteni az alkalmazást. Amint végzett a telepítéssel, elérhetővé válik egy URL segítségével, amit a világ minden pontjáról el lehet majd érni.

5. Célkitűzés

Adathalmaz feldolgozásánál az elsődleges célpont az, hogy lehetőség szerint a legrészletesebben feltárásra kerüljön az elérhető adat és hogy a lehető legpontosabb elörejelzést tudjuk szolgáltatni a felhasználó számára. Ennek a célnak az elérése egy többlépéses folyamat.

1. lépés: Első lépésben szükséges meghatározni a feladatot. Ehhez arra van szükség, hogy az adott témát részletesebben megismerjük. Ha kialakult egy elképzelés, hogy milyen adatok lehetnek érdekesek a témában, akkor folytatódhatnak a munkálatok.

2. lépés: Ebben a lépésben szükséges megtalálni azokat az adatforrásokat, ahonnan az adatok fognak származni. Ezekből a forrásokból az adatokat össze kell gyűjteni és egy közös szerkezetre hozni. Ezek után érdemes ezt egy file-ban eltárolni (csv)

3. lépés: Az összegyűjtött adatokat ebben a lépésben rendkívül fontos megtisztítani, mivel, ha ez a lépés kimarad, a későbbiekben gyakran hamis kiértékelésekhez vezethet.

4. lépés: A megtisztított adatokat amennyiben lehetséges, érdemes bővíteni. Itt olyan transzformációkat lehet elvégezni, amivel plusz olyan oszlopokat hozunk létre, amik hasznosak lehetnek a bányászat során. Ezek lehetnek egy másik adatforrásból érkező információk, vagy a meglévő oszlopok átalakításával vagy módosításával létrehozott új attribútumok.

5. lépés: Ebben a lépésben egy olyan adathalmaz áll rendelkezésre, amiben nagy valószínűséggel már lehet találni valamilyen mintázatot vagy érdekességet a diagramok segítségével. Ezt a lépést a feltáró adatelemzésnek nevezik (Exploratory Data Analysis)

6. lépés: Ebben a lépésben már nincs akadálya annak, hogy valamilyen gépi tanuló algoritmus alkalmazzunk modellépítéshez. Az első lépésben meghatározott feladatokhoz mérten szükséges választani a rendelkezésre álló algoritmusok közül. A gépi tanuló modell létrehozása után, alkalmazhatóak a kiértékelésnél.

7. lépés: Az adatbányászat esetében nem minden esetben sikeres első alkalommal az adatfeltárás vagy vizsgálat. Ezesetben többszöri iterációra van szükség.

A cél a munka során egy olyan gépi tanuló modell építése és alkalmazása, ami képes megjósolni egy futó nevezésének a kimenetelét. A kérdés egyszerű. Képes-e teljesíteni egyénileg egy ilyen hosszúságú szupermaraton-t, vagy felkényszerül adni a versenyt valamelyik nap. Az is vizsgálható, hogy melyik napokon adták fel a legtöbben, esetleg melyik szakasz tekinthető a legnehezebbnek. Ehhez további kutatásra lesz szükség, mivel

az elérhető adathalmazokon belül jelenleg ilyen információk nincsenek jelen, tehát valamilyen külső segítséggel adatbővítésre lesz szükség.

Amennyiben sikerült feltárni azokat az információkat, amiket célul kitűztünk, lehet átültetni a kapott eredményt egy interaktív webes felületre, hogy ezt a kutatást mások számára is elérhetővé tegyük.

6. Megvalósítás

A megvalósítás a tipikus adatbányászati lépésekkel történt meg. Mivel rendelkezésre állt az adathalmaz, nem volt szükség különböző forrásból begyűjteni ezeket, ezért ez a lépés az első iteráció során kimaradt, de egy másik iteráció folyamán szükség volt adatbövítést alkalmazni. Természetesen a korábban már ismertetett cél is kialakult már, emiatt nem kellett ezzel sem huzamosabb időt tölteni. A tényleges munkafolyamatok lépéseinek részletesebb leírását a további fejezetekben lehet megismerni.

6.1. Első iteráció

A munka kezdetével elöször az elérhetö adathalmazt kezdtem feldogozni, ami a 2015 és 2020 –as évek közötti egyéni eredményeket foglalta magába. A kiinduló adathalmaz magas dimenziójúnak tekinthető, mivel számos tulajdonsággal rendelkezik. Tekintsünk is bele, mégis milyen oszlopok voltak elérhetőek a kiindulásnál.

6.1.1. Adathalmaz

Amint már említettem, a kiinduló adathalmaz számos tulajdonsággal rendelkezett alapból. A tulajdonságok között voltak hasznos oszlopok, viszont számos olyat is tartalmazott, amire több mint biztos, hogy nincs szüksége az osztályozó modellnek a tanítás folyamán. Ettől függetlenül, felhasználható adatról van szó így is, amennyiben diagramok segítségével szeretnénk a verseny eredményiről kimutatást vagy egy vizualizációs alkalmazást készíteni.

1. táblázat kiinduló adathalmaz tulajdonságai típusmegjelöléssel

Kiinduló adathalmaz			
Nominális	**Ordinális**	**Folytoson**	**Diszkrét**
Név	Helyezés	Rajtszám	Kategória
Ország	Születési év	Esemény éve	-
Csapat	Napi bontások időadatai	-	-
Város	Napi összesített időadatok	-	-

Nem	Végeredmény	-	-
Rajtszám	Megtett táv	-	-

6.1.2. Adatmigráció

Mivel külön CSV-fájlokban voltak eltárolva az adathalmazok, ezért ezeket egy közös szerkezetre kellett migrálni. Ehhez segítségünkre van a pandas könyvtár. Használatával könnyedén összeadhatóak az adathalmazok, amennyiben az oszlopok és a tulajdonságok megegyeznek a különböző évi adatokkal. A mi esetünkben szerencsére megegyeznek, ezért ezt pár sornyi kóddal végre is tudjuk hajtani. Az így beolvasott CSV fájlokat egy DataFrame-re hallgató adattípusba olvassa be a python amit a pandas már megfelelöen tud kezelni. Ez az adattípus egy olyan 2 dimenziós címkézett adatstruktúra, ami képes különbözö típusú oszlopokat tárolni. Úgy is lehet rá tekinteni, mint egy egyszerü SQL táblára, vagy egy dictionary-re, ami sorozatobjetumokat tartalmaz.

6.1.3. Adattisztítás és transzformáció

Sajnos a való életben nem gyakori az az eset, amikor a feldolgozandó adathalmaz olyan állapotban van, hogy hozzá sem kell nyúlni, hanem egyből rá lehet engedni a gépi tanuló algoritmusokat. A valóság ennél egy picivel gyötrelmesebb. Előfördulhat, hogy az adatok nem konzisztens módon lettek rögzítve. Ez történt a használt adathalmaz esetében is. Sok hiányos adattal rendelkezett, amit így nem lehetett felhasználni az első iterációban. Nem tartalmazott olyan lényeges oszlopokat sem alapból az adathalmaz, amire mindenképpen szükségünk lett volna az osztályozás során, ezért ezt saját kezűleg kellett hozzádni. Ez a tulajdonság azt jelöli, hogy az egyén az adott évben bejefejezte-e a versenyt. Ezt olyan módon tudjuk utólagosan hozzádni az adathalmazhoz, hogy a megtett kilóméterből következtetjük, hogy ki az aki lefutotta mind a 198 kilómétert és ki az aki nem.

A megtett kilóméter és az eredmény oszlopokat felhasználva kitudjuk számolni a futó átlagsebességét is, így ismét egy feltételezhetöen fontos tulajdonsággal tudjuk bővíteni az adathalmazt.

6.1.4. Adatdiszkretizáció

Az adatdiszkretizáció során a cél az, hogy a folytonos adatokat valamilyen kategórikus adatszerkezetre tudjuk hozni.

Vödrözés

Egy olyan eljárás, amivel a folytonos értékeket tetszőleges *n* számú vödörbe osszuk fel. Két kiemelendö módja van ennek az eljárásnak.

- Azonos elemszámú vödrözés
- Azonos intervallumú vödrözés

Az eljárás nagyon egyszerű és hatékony. Először is fogalmazzuk meg, hogy hány kategóriába szeretnénk besolorolni az adathalmazt. Ezt fogjuk jelölni a vödrök számával. Azonos elemszámú vödrök esetében, minden vödör pontosan ugyan annyi eredményt tartalmaz, így azonos eloszlásúak lesznek a kategória címkék. A másik eljárás, viszont az intervallumot osztja fel azonos részekre, és az eredmények által felvett értékek alapján fogja valamilyen kategóriába besorolni az adott sort. Így jól látható, hogy előfordulhat, hogy nem lesz azonos eloszlásban az így kapott eredmény. Ehhez segítségünkre van a már említett pandas könyvtár. Gyakorlatilag egy sornyi kóddal átalakítható egy folytonos érték kategórikussá. Az azonos intervallomú verzióval az alábbi módon végezhető:

```
bszm['sebesseg_kat']=pd.cut(bszm['atlag_tempo'],5,
labels=['gyors','koz_gyors','atlagos','koz_lassu','lassu'])
```

A ***bszm*** a kódban a DataFrame-et jelöli. Ehhez úgy tudunk egy új oszlopot hozzádni, hogy szögletes zárójelbe rakjuk az oszlop nevét. Ehhez az oszlophoz hozzárendeljük az értéket, amit a pandas könyvtár *cut* metódusa hoz létre. Az azonos elemszámú vödrök létrehozásához, a pandas ***qcut*** metódusa alkalmazható.

```
bszm['sebesseg_kat']=pd.qcut(bszm['atlag_tempo'],5,
labels=['gyors','koz_gyors','atlagos','koz_lassu','lassu'])
```

6.1.5. Feltáró elemzés

Ahhoz, hogy nagyobb rálátásunk legyen arra, hogy mégis milyen információkat tartalmaz az adathalmaz, ahhoz a feltáró elemzésre lesz szükség. Fontos lépése az adatbányászati folyamatoknak, mivel így jobban értelmezni tudjuk azt, hogy az adatok

hogyan viszonyulnak egymáshoz képest. Sok esetben ezen a ponton fényt lehet deríteni esetleges inkonzisztenciákra vagy hibákra. A különbözö diagramok használatával outlier-ek felfedezésére is lehet használni ezeket az eljárasokat. Rengeteg fajta feltáró elemzést segítő könyvtár áll rendelkezésre, ezekre példa:

- Matplotlib,
- Plotly,
- Seaborn,
- ggplot,
- Altair
- DataPrep

Valójában a plotting könyvtárak segítségével tudunk a legalkalmasabban feltáró elemzéseket készíteni. Alkalmazásuk egyértelmű, néhány függvény segítségével felparaméterezhető a diagram, amit a python futtatva megjlelenít a jupyter notebook felületén, ami akár exportálható is egyből a felületen.

6.1.6. Osztályozás

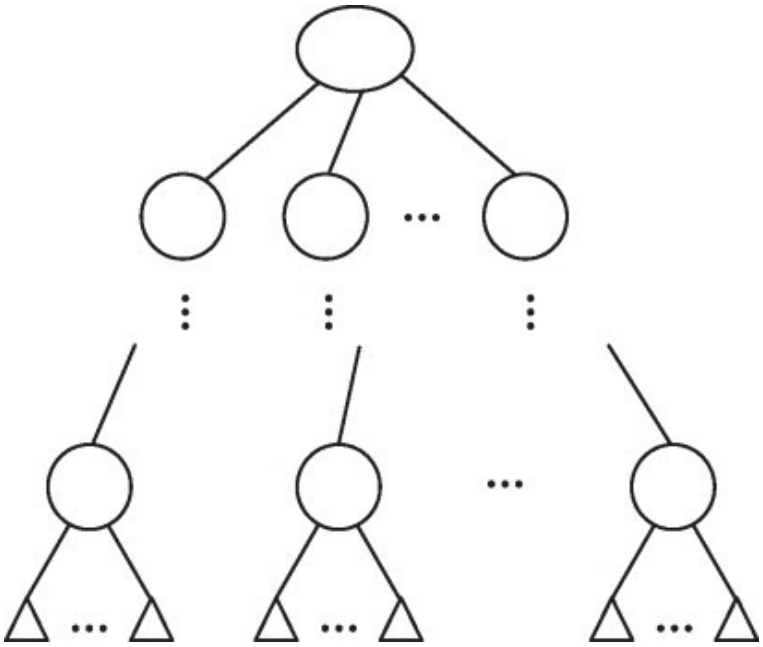
Az elsö iterációs eljárás során két osztályozó eljárást alkalmaztam. Ezek a következők voltak:

- Döntési fa osztályozó,
- Random forest osztályozó

Elsősorban azért ezekre az algoritmusokra esett a választásom, mivel alkalmazásuk gyakori és általában jó eredményeket tud hozni a megfelelő adathalmazokon.

6.1.6.1. Döntési fa

A döntési fa osztályozónak a müködése meglehetösen egyszerű. Fa szerűen lehet a segítségével tudást ábrázolni, amely akár átírható egy diszkrét szabályhalmazzá is, így könnyítve tovább a megértését. A fő előnye az, hogy képes különböző részhalmazokat használni az osztályozás különböző szakaszaiban. Egy általános döntési fa modell mindig egy csomópontból indul ki. Természetesen minden osztályozó modellnek vannak előnyei és hátrányai, ezért érdemes tudni azt, hogy milyen esetben melyik algoritmus alkalmazása lesz a legoptimálisabb. [4] [5] [6]



2. ábra Döntési fa

Előnyei

- Könnyen értelmezhető,
- Numerikus és kategorikus adattal is tud dolgozni
- Nincs sok előfeltétele (pl.: nem szükséges dummy változókat létrehozni vagy vödrözést alkalmazni)

Hátrányai

- Könnyedén túlilleszt

6.1.6.2. Random forest

A random forest egy olyan továbbfejlesztése a döntési fa osztályozónak, ami növelni tudja a teljesítményt. Ezt oly módon teszi, hogy a modell létrehozásánál, több döntési fát hoz létre. Ezeknek a fáknak a számát általában *n* változóval szokták jelölni, és tetszőleges pozitív egész számot vehet fel. Az osztálycímkét végül ezen egyes döntési fáknak az eredménye fogja adni, úgy, hogy a többségben lévő osztálycímke lesz a jósolt érték.

Előnyei

- Mind osztályozásra és regresszióra is alkalmazható
- Szintén alkalmas numerikus és kategórikus adatokkal is
- Feature selection automatikus, minden tulajdonságra épít fát
- Hasznos magas tulajdonságszámú adathalmazokon.

Hátrányai

- Nagy adathalmazokon lassú
- Fekete doboz szerű működés, nem lehet nagyban befolyásolni a működését

6.1.7. Kiértékelés

A két osztályozó algoritmus alkalmazásával létrehozott gépi tanuló modellek sajnálatos módon nem tudtak optimális pontosságot elérni. A tanított modell többszörös tanítási eljárás során 60 és 65 százalék közötti pontosságot tudot eredményezni. Ebből jól látható, hogy ez éppen hogy pontosabb a tippelésnél. Ez feltételezhetően amiatt van, mert nem megfelelően volt előkészítve az adathalmaz a gépi tanuló eljárásoknak, ezért mindenképpen szükséges lesz egy második iteráció és az adathalmaz részletesebb előkészítése. Emelett az adathalmaz bővítése is cél a következö iterációban, ezért az esemény szervezőivel szükséges felvenni a kapcsolatot. [7]

6.2. Probléma

Mivel első iteráció eredményei nem voltak kielégítőek, ezért szükséges volt az adathalmaz részletesebb, több oldalról történö megvizsgálása és a fennálló problémáinak elemzése.

6.2.1. Kis adathalmaz

A kis adathalmaz minden esetben problémát jelent adatbányászati eljárások során és nehézzé teszi az osztályozó modellek pontos teljesítőképességét. [8]

Túlillesztés

A túlillesztés gyakori hiba szokott lenni a legtöbb adatbányászati projekt során. Olyan esetekben fordul elő, amikor a gépi tanuló modell, túlságosan is rátanul egy olyan értékre, amely valószínűleg nem reprezentálja a pontos kimetelt. Az ilyen túlillesztés gyakori kis adathalmazokon, mivel a jellegzetes minták hiányoznak, ezért a tanuló algoritmusok akkor is finomítják tovább a modellt, amikor már csak néhány elem maradt a tanítóhalmazban.

Emiatt egyértelmű volt, hogy növelni kell az adatmennyiséget. Ahhoz, hogy ez kivitelezhető legyen, a versenyszám szervezőivel kellett felvennem a kapcsolatot és megérdeklődni, hogy rendelkezésre állnak-e a 2015 előtti versenyek adatai is. Hamar érkezett is válasz, amiben megkaptam az összes általuk szervezett eseménynek az eredményeit, egészen a 2008-as évig visszamenőleg, így már majdnem kétszer annyi adatot lehetett a munka során feldogozni.

6.2.2. Kiegyensúlyozatlan adathalmaz

A kapott adathalmazt összeolvasztva az előzölegesen elérhetö adathalmazzal, már közel 2500 sornyi adattal rendelkezünk, azonban az adathalmaz növekedésével nőtt az adathalmaz kiegyensúlyozatlansága is. [9]

Előzölegesen is jelentősen egyenlőtlen eloszlása volt az osztálycímkéknek, viszont a 2008 és 2011 közötti eredmények ezt jelentősen fokozták. Az említett évek eredményi csak azon futók eredményeit tartalmazták, akik befejezték a versenyt, ebből adodóan az osztályozó modell épitésénél használhatatlanok lesznek. A teljes adathalmazon az osztálycímkék eloszlása 25 – 75 százalékos, ahol a 25 százalék jelöli a versenyt feladókat. Ez ismétetlten a egy olyan hamis modell teljesítményt eredményez, ami “high bias”, így majdnem minden modell 90 és 95 százalékos pontosságú lett, ami nem valós pontosság, hanem túlságosan rátanult a jelentős többségben lévő osztálycímkére. [7]

6.2.3. Zajos adathalmaz

A különböző évek közti adatoknál magas volt az inkonzisztencia, ezért rengeteg zajt is tartalmazott. Majdnem minden évben különböző módon rögzítették az adatokat, ezért szükséges volt az adathalmaz teljes körű megvizsgálása és tisztítása. A versenyek hossza váltakozó volt, ezért az alapvető osztálycímke generálásnál, sokat hamisan generáltunk a tulajdonságbővítésnél.

6.3. Második iteráció

A második iterációban a fent említett problémákat figyelembe véve folytattam a munkálatokat. Az irodalomkutatással rengeteg megoldást és megközelítést lehetett találni a problémákra, mivel ezek sok esetben előfordulhatnak, ezért számok publikáció

foglalkozik kifejezetten ezeknek a kiküszöbölésére és a gépi tanuló algoritmusok teljesítményének javítására.

6.3.1. Adathalmaz bővítése

Az adathalmaz bővítése a szervezők megkeresésével vált lehetővé. Az adatokat egészen 2008-ig visszamenőleg szolgáltatták, amivel közel 2500 sort is elérte az összesített adathalmaz. Ebből, ahogyan előzőleg említettem 3 évnyi adat nem volt felhasználható az osztályozás során, mivel növelte az osztálycímkék egyenlőtlenségét.

Az adathalmaz bővítését természetesen nem feltétlenül csak a sorok növelésével lehet megoldani, hanem a tulajdonságok kiegészítésével is. Visszakereshető az eseményeknek a pontos napja, így egy másik weboldal segítségével összegyűjthetőek a napi időjárás adatok egészen a 2009-es évig. Mivel az adathalmazunk tartalmazza a 2008-as adatokat is, így ehhez olyan módon kerültek kiegészítésre az információk, hogy az átlagát vettük az összes többi adatnak. Ez nem a lehető legoptimálisabb eljárás, viszont legalább egységesen mindenhol tartalmaz adatot az oszlop. A részletes időjárás adatok olyan információkat tartalmaznak, mint hőmérséklet, felhősség, légnyomás, eső mértéke, szél erőssége és széllökések erőssége. [10]

A verseny információs oldaláról begyűjthető a verseny útvonala, a napi bontások és szakaszok hossza. Ezekkel is bővíthető az adathalmaz, így még több olyan tulajdonsággal rendelkezik, amik hasznosak és érdekesek lehetnek az adatvizualizációs felület készítésénél.

2. táblázat bővített adathalmaz tulajdonságai típusmegjelöléssel

Bővített adathalmaz			
Nominális	**Ordinális**	**Folytoson**	**Diszkrét**
Név	Helyezés	Rajtszám	Kategória
Ország	Születési év	Esemény éve	***Hőmérséklet***
Csapat	Napi bontások időadatai	-	***Tempó kategória***
Város	Napi összesített idöadatok	-	***Felhősség***
Nem	Végeredmény	-	***Légnyomás***
Rajtszám	Megtett táv	-	***Eső mértéke***

-	***Napi szakaszok hossza***	-	***Szél***
-	***Napi szakasz tempók***	-	***Széllökések***
-	***Napi tempók***	-	-
-	***Időjárás adatok***	-	-

6.3.2. Adattisztítás és transzformáció

A második iterációban, az adattisztítási eljárások során, sokkal részletesebben került megvizsgálásra az adathalmaz. Pontosabban lett a célosztálycímke legenerálva, mivel eddig a megtett kilométert vizsgálta a program, ami helytelen eredményeket hozott, mivel a verseny hossza évről évre változott. A legegyszerűbben olyan módon lehetett pontosan legenerálni, hogy a helyezését vizsgáltuk meg az egyéneknek. Amennyiben volt helyezése, befejezte a versenyt, amennyiben *NaN* értéket vett fel, abban az esetben feladta.

A születési dátumot felhasználva létrehozható a kor tulajdonság, amit úgy tudunk pontosan megkapni, hogy kivonjuk az esemény évéből a születési évet.

Az összes időadat átdolgozásra került másodperc formátumban, mivel a python nem volt képes dolgozni az időadat formátummal, mivel *string* adattípusként dolgozta fel. Az átlagos tempó kategórikus adatot olyan módon számoltuk ki, hogy az eredményt elosztottuk a futott kilométerrel. Általános megjelenítése a tempónak futás esetén a perc / kilométer, tehát, hogy általánosan mennyi időbe telt egy kilométer lefutása a futó számára.

Mivel a kor/nem kategória sem volt egységes, ezért ezeket teljesen újra generáltam egy közös rendszerre, ahol az alábbi intervallumokat tekintjük a különböző csoportoknak.

3. táblázat Korcsoportok

Korcsoportok					
	Csoport 1	**Csoport 2**	**Csoport 3**	**Csoport 4**	**Csoport 5**
Férfi kor	18 - 27	28 - 39	40 – 50	51 - 59	60+
Női kor	18 - 27	28 - 39	40 - 55	55+	-

6.3.3. Outlier analízis

Az outlier analízis az adattisztítás egy kiegészítő lépése, amire azért van nagy szükség, hogy a kitűnően gyanús értékek eliminálásra kerüljenek. Ennek a lépésnek a hiányában problémákat okozhat a gépi tanulás során. [3]

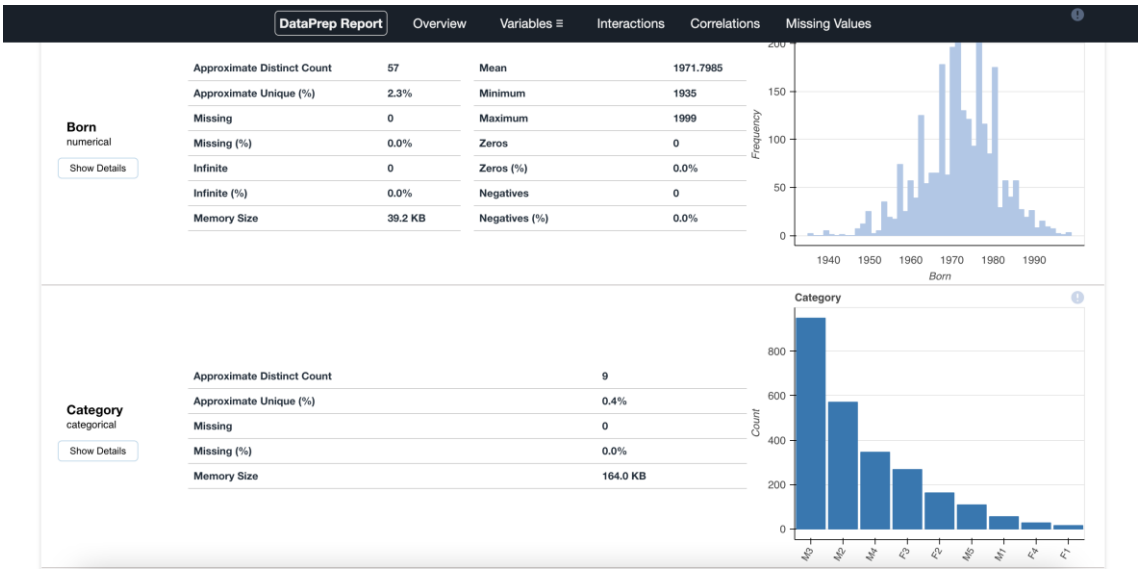
6.3.4. Feature engineering

A feature engineering alatt egy olyan eljárást értünk, amikor a meglévő tulajdonságokat alakítjuk át olyan módon, hogy egy újat hozunk létre, ami hasznos lehet az osztályozási modell építésénél. Ahhoz, hogy ezt optimálisan lehessen felhasználni, szükséges a témakörben való jártasság. A motiváció mögötte az, hogy javítsák a gépi tanulás folyamat eredményeinek minőségét, ahhoz képest, mintha a nyers adatokat használná fel az algoritmus.

6.3.5. Feltáró elemzés

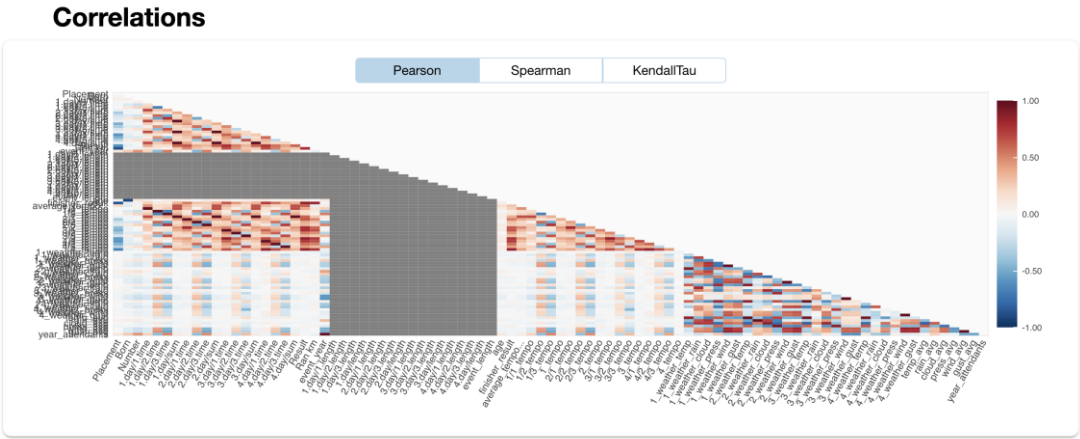
A feltáró elemzés a második iterációban részletesebben került alkalmazva. Ehhez egy nagyszerű könyvtár állt rendelkezésre, ami a DataPrep névre hallgat. Használatával gyakorlatilag egyenesen az adathalmazból készít egy automatizált kimutatást, rengeteg hasznos információval.

Az adattípusoknak megfelelően, automatikusan választja ki a megjelenítendő diagrammokat, amiket a legrészletesebb módon, minden információval ellát. Ezeket a riportokat megjeleníthetjük egyenesen egy Jupyter Notebook-ban, vagy kiexportálhatjuk egy HTML szerkezetű fájlba is, így könnyítve a vizualizációt.



3. ábra DataPrep riport

Jól megjeleníti azokat az oszlopokat is, amelyek üres értékeket tartalmaznak, így segíti az adattisztítás menetét is. A tulajdonságok között figyeli a korrelációs értékeket is, amik megtekinthetőek mind táblázatosan.



4. ábra DataPrep korreláció kiértékelés

6.3.6. Importance analysis

Az importance analízis egy olyan eljárás, amit az osztályozást megelőzően érdemes végrehajtani. Használatával megtudhatjuk, hogy melyik tulajdonságok lesznek a legfontosabbak a gépi tanuló modell létrehozásánál.

6.3.7. Tulajdonság kiválasztás

A tulajdonság kiválasztás, másnéven feature selection, egy szintén kiemelkedően fontos lépése az adatbányászati folyamatnak. Ezen a ponton kell kiválasztani, azokat a tulajdonságokat, amiket az algoritmus használni fog az osztályozás során. Fontos, hogy modellépítésnél egymástól független tulajdonságokat válasszunk, ki, mivel ennek hiányában hamis vagy rossz eredményeket adhat eredményül a prediktív modell.

Amennyiben magas számosságú tulajdonsággal rendelkezik az adathalmaz, akkor a bemeneti változók számát érdemes csökkenteni, hogy így a számítás költsége csökkenjen, esetlegesen a modell teljesítménye növekedjen.

6.3.8. Keresztvalidálási eljárások

Keresztvalidálási eljárásoknak azokban az esetekben van nagy haszna, amikor az adathalmazon az általánosan használt tanító-teszt-vágás bizonytalan eredményeket hoz a modellépítés során. Bizonytalan eredmények alatt olyan változó modell pontosságokat értünk, amik széles intervallumon vesznek fel értékeket. Ez olyan esetben fordul elő, amikor a kétharmados vágással nem megfelelően oszlanak el az osztálycímkék a tanító, teszt és validációs halmazokban. Ennek a bizonytalanságnak a kiküszöbölésére alkalmazzuk a különféle eljárásokat.

Számos eljárás létezik, azonban az adathalmaz válogatja, hogy melyik eredményezi a legoptimálisabb működést. A működésük azonban mind hasonló, mivel az összes eljárás bizonyos *fold*-okon, dolgozik, ami nem jelent mást, mint hogy az adathalmaz hány részre kerül felosztásra a validálás és a tanítás során. A dolgozatban három módszer kerül bemutatásra, amelyeket alkalmazva javíthatóvá vált az osztályozó modell teljesítőképessége. [11]

Leave One Out Cross Validation

A legegyszerűbb keresztvalidálási eljárás, amit inkább régebben alkalmaztak. Egy olyan speciális esetű keresztellenőrzés, amiben a foldok száma megegyezik az adathalmaz példányainak számával, ezáltal a tanuló algoritmus minden egyes példánya egyszer kerül alkalmazásra, úgy, hogy az összes többi példányt tanító készletnek tekinti, az egy darab kiválasztott elemet pedig egy egyelemű teszthalmaznak használja.

Előnyei

- Low Bias, tehát nem tanul rá egy osztálycímkére, mivel minden elemszámra validációt hajt végre
- Megbízható

Hátrányai

- Nagy elemszámú halmazokon lassú,
- Nagy költségű,
- Elavult

K-fold Cross Validation

A K-fold Cross Validation egy olyan keresztvalidációs eljárás, ami egy paraméterrel rendelkezik, ami a *k*. Ez a változó arra utal, hogy az adathalmazt, hány különálló részre bontjuk fel az eljárás során. Miután felbontottuk az adathalmazt egy iterációs eljárással végig haladunk a vágott adathalmazokon, úgy, hogy minden iterációban egy azonos méretű teszthalmazt és tanító halmazt alkalmazunk. Az iterációkon haladva, minden alkalommal egy újabb teszt halmaz lesz felhasználva.

Ennek az eljárásnak az egyszerűbb megértésének a kedvéért vegyünk példának egy 1000 elemű adathalmazt és fogalmazzuk meg a *k* változónak az értékét 5-re.

4. táblázat K-fold Cross Validation minta

Osztálycímkék elolszása	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Pontosság**
180 / 20	200	800				80.12 %
198 / 2	200	200	600			84.94 %
120 / 80	400		200	400		76.42 %
110 / 90	600			200	200	70.32 %
200 / 0	800				200	92.02 %

Így jól látható, hogy 5 iterációt fogunk végrehajtani a keresztvalidálás során. Az első foldnál például vesszük az adathalmaz első 200 elemét a teszthalmaznak, a maradék 800 elemet pedig használjuk tanító halmazként. A második foldnál vegyük a soron következő 200 sor utáni ismét 200 elemet és a maradék 800-at alkalmazzuk a tanító halmaznál. Az iteráció így folytatódik egészen addig, amíg minden elem legalább egyszer

nem került be egy tanítási halmazba. Amint ez megvan, megkapjuk az egyes iterációk pontosságának az eredményét, amiknek az átlagát véve megkapjuk a valós pontosságát a tanított modellnek.

Előnyei

- Növeli a teljesítményét az osztályozásnak,
- Ismeretlen adathalmazokat segít jobban feltárni,
- Kis adathalmazokon optimális

Hátrányai

- Kiegyensúlyozatlan adathalmazokon még mindig fennáll az esélye arra, hogy túlságosan is rátanul egy osztálycímkére.
- Nagy adathalmazokon lassú

Stratified K-fold Cross Validation

A Stratified K-fold Cross Validation esetén egy olyan keresztvalidációs eljárást értünk, ami az előzőleg bemutatott K-fold Cross Validation hibáit küszöböli ki. A működésük nagyban megyegezik, viszont ez már képes pontosabb eredményeket mutatni a tanítás során. Ezt úgy képes elérni, hogy a különböző iterációkban, a teszthalmazban a teljes adathalmazhoz hasonló osztálycímke eloszlását biztosítja, így nem fordulhat elő olyan fold, amiben erőteljesebb kiegyensúlyozatlanság van jelen. Ezt a működést az ábra segítségével tesszük könnyebben érthetővé.

5. táblázat Stratified K-fold Cross Validation minta

Osztálycímkék elolszása	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Pontosság**
140 / 60	200	800				77.83 %
140 / 60	200	200	600			72.32 %
140 / 60	400		200	400		77.01 %
140 / 60	600			200	200	78.23 %
140 / 60	800				200	74.52 %

Ahogy látható, minden iterációs lépésben, a teljes adathalmazhoz hasonló eloszlásban lesznek jelen az osztálycímkék, így elősegítve a lehető legpontosabb

kiértékelését a tanulási folyamatnak. A kiértékelés az elődjéhez hasonló módon történik, az iterációk pontosságának az átlagával.

Előnyei

- A valós pontosságot tovább növeli az osztályozásnak,
- Minden iterációban azonos eloszlású osztálycímkék
- Kis adathalmazokon optimális

Hátrányai

- Lassú

6.3.9. Osztályozás

Az első iterációhoz képest, ebben a lépésben többféle osztályozó algoritmus is alkalmazva lett, így lehetőség nyílt a különféle modellek kiértékelésére és teljesitőképességének felmérésére. [12]

6.3.9.1. Döntési fa

Mivel egy pontosabb és jobban tisztított adathalmazzal lehetett folytatni a munkát a második iterációban, ezért lehetőség nyílt egy pontosabb modell építésére is a döntési fa osztályozó alkalmazásával. Az így kapott eredmény többszöri tanítás után, 70-72 százalék körüli eredményt produkált.

6.3.9.2. Random forest

A random forest osztályozó esetében hasonló eredményeket lehetett kapni. Ezzel az algoritmussal egy fokkal jobb eredményeket születtek, ám ez továbbra is nem a legoptimálisabb pontosság. Ez a szám a 71 és 73 százalék körül mozgott többszöri tanítás során.

6.3.9.3. SVM

A Support Vector osztályozó az előző iterációban nem volt alkalmazva az adathalmazon, viszont az irodalomkutatásban többször is ajánlották az alkalmazását kisebb méretű és nagy tulajdonságszámú adathalmazokon. Működése ennek az algoritmusnak szintén egyszerű és meglepően alkalmas kis adathalmazokra. Olyan

esetekben is képes jól teljesíteni, ahol a tulajdonságok száma több mint az adtasorok nagysága. Jól alkalmazható outlier detektálásra és regressziós eljárásokra is egyaránt. Alkalmazásával az adathalmazon jobb eredményt hozott, mint a döntési fa osztályozók és pontosságuk 74-76 százalékos pontosságot produkált. Az eddig vizsgált gépi tanuló algoritmusokból ez teljesített a legjobban. [13]

6.3.9.4. Neurális háló

A neurális háló osztályozó is alkalmazva lett az adathalmazon, bár a teljesítményük egyáltalán nem hozott jobb eredményeket az előző modellekhez képest, ezért nagyobb hangsúlyt fektettem a többi algoritmus működésének az optimalizálására és kiértékelésére.

6.3.9.5. Naive Bayesian

A Naive Bayesian osztályozó egy olyan eljárás, ami valószínűségeket használ fel az osztályozó feladatok ellátására. A többi felügyelt gépi tanuló algoritmusokhoz hasonlóan, a Naive Bayesian szintén célváltozók előrejelzésére törekszik. Legnagyobb különbség, hogy ez az eljárás azt feltételezi, hogy a tulajdonságok teljes mértékben függetlenek egymástól, és nincs összefüggés a címkék között. Ez viszont a valóságban nem mindig így van, és csak egy feltételezés, hogy a tulajdonságok nem korrelálnak egymással, innen ered a naív elnevezése az eljárásnak.

Az eljárás a Bayes elven alapszik, amí egy feltételes valószínűséget számol ki. Ez a valószínűség egyszerűen egy esemény bekövetkezésének valószínűségét jelenti, és mindig 0 és 1 közötti értéket vesz fel. Az *A* esemény valószínűségét *p(A)*-ként jelöljük, és úgy számítjuk ki, hogy a kívánt eredmény számát elosztjuk az összes kimenet számával.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{1}$$

Mivel számos kategórikus adattípusú tulajdonsággal rendelkezik az adathalmaz, ezért érdemes ezt az oszályozási eljárást alkalmazni. Az összes tesztelt gépi tanuló modell közül, ennek az alkalmazásával lehetett a legoptimálisabb eredményeket elérni.

Előnyei

- Gyors,
- Kategórikus adatokra jól alkalmazható

6.3.10. Kiértékelés

A különböző osztályozók alkalmazásával jól láthatóvá vált, hogy az adathalmazra melyek azok, amik a legjobb teljesítményt tudták nyújtani.

6. táblázat gépi tanuló algoritmusok összehasonlítása

Mért átlagos pontosságok	
Osztályozó	**Pontosság**
Döntési fa	70 – 72 %
Random Forest	72 – 74 %
Support Vector Machine	73 -75 %
Neurális háló	63 – 65 %
K-Nearest-Neighbor	68 - 70 %
Naive Bayesian	76 - 77 %

7. Alkalmazás

Miután sikerült a célul kitűzött feladatot megvalósítani, ezt valamilyen módon szeretnénk elérhetővé tenni egy alkalmazás, weboldal segítségével. Sokfajta megközelítése lehet egy ilyen webapplikáció fejlesztésének, de a dolgozat elkészítésénél, törekedtem az egyszerű alternatívák használatára. Emellett az is fontos volt, hogy egy modern, letisztult felület készüljön el, ahol szépen, egyszerűen vizualizálva legyenek a különböző táblázatok és diagramok alkalmazásával a rendelkezésre álló adat. Természetesen elengedhetetlen manapság, a felhasználóbarát feületre való törekvés is, és a könnyü és gyors elérhetőség.

Ahhoz, hogy a fent említett elvárásoknak mind megfeleljen az alkalmazás, a streamlit névre hallgató python könyvtárt fogom felhasználni.

7.1. Telepítés

A fejlesztés megkezdésénél szükséges volt először is a használt programozási nyelv telepítése az eszközre. Tetszőlegesen választható a python verziójának a száma, de ajánlott mindenképpen a Python3 és ennél újabb verziók, mivel a 2-es verzió már jelentősen elavultnak tekinthető, és rengeteg hasznos funkciót nem tartalmaz, ami könnyíteni tudja a fejlesztések menetét.

Ahhoz, hogy a adatbányászati munkát tudjunk végezni, elöször is telepíteni kell a már az elöző fejezetben is említett könytárakat. Ezeket a python alapértelmezett csomagkezelőjével lehet telepíteni.

```
python3 -m pip install -r requirements.txt
```

Ahhoz, hogy ne kelljen a különböző könyvtárakat egyesével telepíteni, használható egy egyszerű szöveges fájl. Annyi dolgunk van vele, hogy a fájlban sorokra szedve beleírjuk a telepítendő könyvtárak nevét, így a parancs futtatásával a csomagkezelő automatikusan végigiterálva a sorokon telepíteni tudja az összes felsorolt elemet.

7.2. Konfiguráció

Miután telepítettük a pythont, szükséges ennek a konfigurálása valamilyen kódszerkesztőben vagy valamilyen integrált fejlesztési környezetben. Ahhoz, hogy a

python-t használni tudjuk a Visual Studio Code-on belül, ahhoz telepíteni kell az ehhez készített bővítményt és be kell állítani a python elérési helyét. Ha az megvan, futtathatóvá válnak az alkalmazśon belül a python scriptek.

7.3. Streamlit

A streamlit keretrendszer működése és használata meglehetősen egyszerű és egy bizonyos szintig könnyedén konfigurálható. A működését egy beágyazott webkeretrendszer szolgálja ki, ami egy webszervert üzemeltet. A webalkalmazást egy böngésző segítségével lehet elérni, ahol egyből megjelenik a kezdőoldal. Működése egy Single Page alkalmazásához hasonló, ahol nincsenek különböző végpontok, csak a főoldal. Az alkalmazásban különböző állapotok vannak, melyek változására módosíthatóak a megjelenített komponensek és oldalak. Így könnyedén hozhatóak létre külön oldalak, nézetek és komponens csoportok.

7.4. Futtatás

Alapkonfigurációjában a könyvtár lokálisan a *localhost* elérhetőségén indítja el a webszervert a 8501-es porton az alábbi terminálparancs kiadásával.

```
streamlit run /...path/filename.py
```

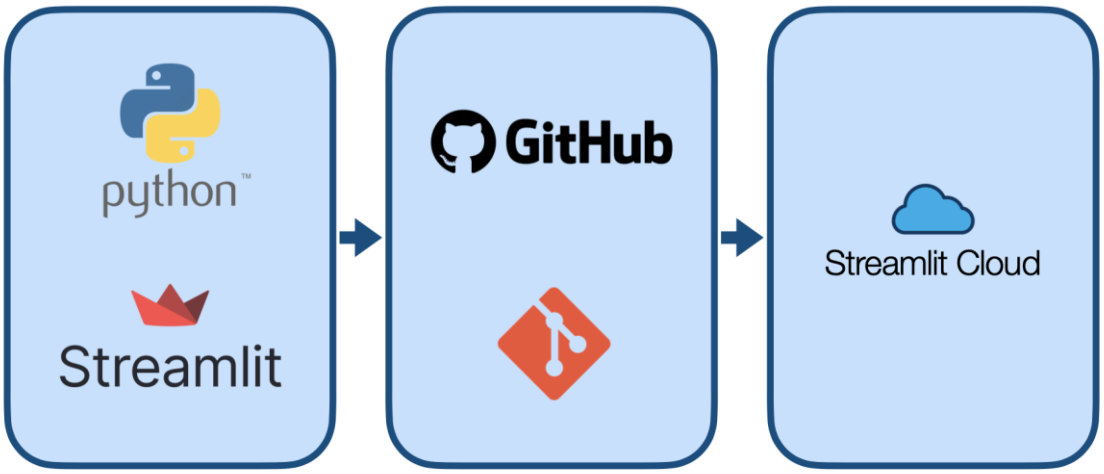
Miután elindult az alkalmazás, automatikusan megnyílik az alapértelmezett böngészőben az alkalmazás.

7.5. Felhő

A streamlit szolgáltat egy ingyenes felhőszolgáltatást is a regisztrált felhasználók számára bizonyos korlátozásokkal. Egyszerre csak 3 alkalmazást lehet futtatni ingyenesen, amennyiben ennél többet szeretne a felhasználó használni, ahhoz a pro előfizetésre van szükség.

Használata a felhő szolgáltatásnak meglehetősen egyszerű, mivel csak egy publikus Github tárhelyre van szükség a beüzemeléshez. Fontos, hogy publikus legyen, mivel csak így tud hozzáférni a streamlit a forráskódhoz. Miután ezt a Streamlit Cloud felületen beállítottuk, rögzíti ezt a felhő rendszerbe és elkezdi telepíteni az alkalmazást a streamlit által szolgáltatott platformra. Innentől bármilyen változást érzékel a Githubon a streamlit,

automatikusan újraindítja és telepíti a webszervert is, így mindig a legfrissebb verziót tudja szolgáltatni a felhasználók számára.



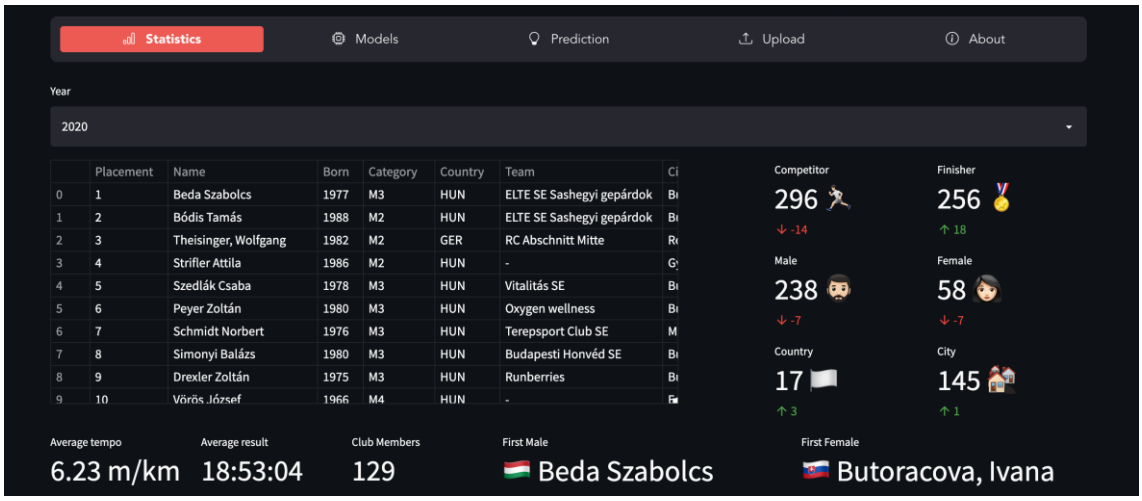
5. ábra Felhő platform felépítése

7.6. Felületek

Az alkalmazás 5 különböző felületből épül fel, amik között a navigációt egy szabad forráskódú könyvtárt felhasználva valósítottam meg. Erre azért volt szükség, mivel a streamlit nem tartalmazott olyan navigációs menüszalagot, amely hasonló módon működött volna, így egy rövid kutatás után találtam ezt az alternatívát. Ennek a könyvtárnak a neve *streamlit-option-menu*.

7.6.1. Statisztika

Ez a felület az alkalmazás legfontosabb felülete, mivel itt tekinthetőek meg a legrészletesebb módon az eredményeknek az adathalmazai. Beépített streamlit komponenseket alkalmazva könnyedén létrehozható volt a felület. Az oldal tetején elhelyezkedő beviteli mező egy szűrőként funkcionál, amiben kiválasztható az esemény éve, vagy az összes esemény. Az itt tárolt érték változtatására az oldalon megjelenő adatok is dinamikusan változnak.



6. ábra Statisztikai felület 1

Hasonló elven működik az oldalon lejjebb található napi kiértékeléseket tartalmazó táblázat és a naphoz tartozó további adatok. Itt a beviteli mezőben a nap száma változtatható, amivel ismételten szűrhető a megjelenítendő adat.

7. ábra Statisztikai felület 2

7.6.2. Modellek

A modellek felületen betekintést nyerhetünk az alkalmazott gépi tanuló algoritmusokba és ezek működésébe.

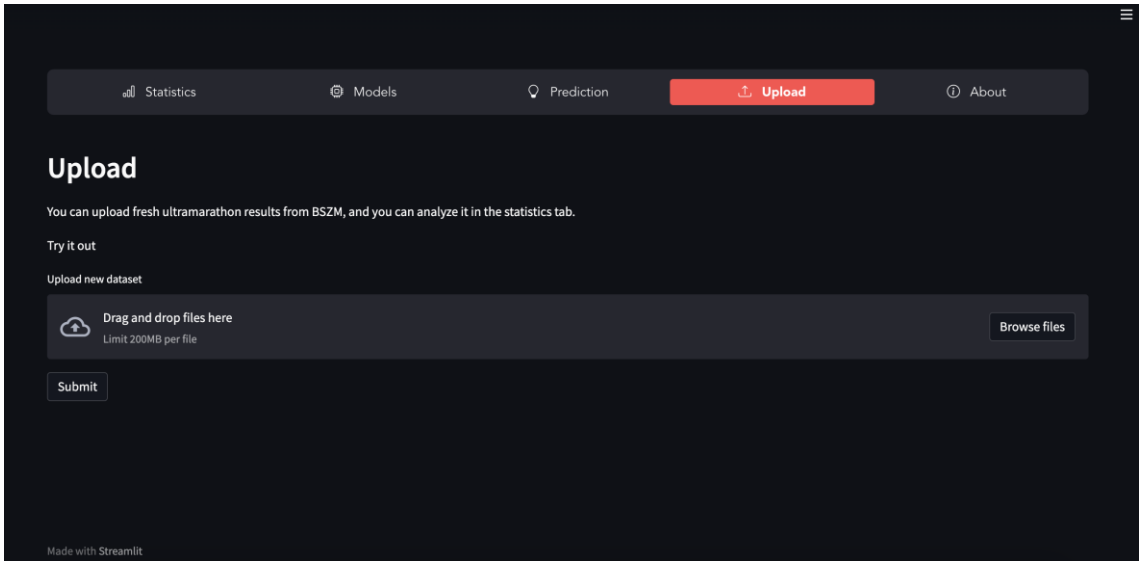
7.6.3. Előrejelzés

Ezen a felületen lehetősége van a felhasználónak az osztályozó modell használatára. Bevitt adatok alapján könnyedén és gyorsan választ tud adni az alkalmazás, miszerint le tudja a futni ezt a versenyszámot anélkül, hogy feladja, vagy felkényszerül adni.

Amennyiben pozitív eredményt jelez vissza a gépi tanuló modell, akkor a rendszer automatikusan jelzi ezt a felhasználónak egy rövid streamlit keretrendszeren belűl implementált animációval.

7.6.4. Adathalmaz feltöltés

Az adathalmaz feltöltés felületen lehetőséget nyújt az alkalmazás új évi eredményeknek a feltöltésére. Mivel javarészt a legfrissebb adatok szerkezete megegyezik, lehetőség nyílik arra, hogy egy új eseménynek az eredményeit automatikus módon feldolgozza a rendszer egy fájlfeltöltés esetén. Az ezen a ponton feltöltött adathalmaz az előzőlegesen is használt adattisztítási és adatbővítési lépéseken megy keresztül, így egységesen tartva a megjelenítendő táblázatokat és diagramokat. Ezen új eredmények mentésre kerülnek az alkalmazást futtató rendszeren, így a következő modellépítés során, ezek felhasználásával további teljesítménynövekedést lehet produkálni a gépi tanuló algoritmusoknál.



8. ábra Adathalmaz feltöltése felület

7.6.5. Információk

Ezen a felületen magáról a Balaton Szupermaratonról és a kutatásról lehet olvasni. Ezek mellett érdekességeket lehet olvasni a munka folyamatáról és a fennálló problémákról és a motivációkról. Mivel a dolgozat befejeztével még nem történt meg a részletesebb együttműködés a szervezőkkel, ezért erről nem lehetett ezen a ponton részletesebb információkkal kiegészíteni az információs felületet.

8. Összesítés

Az adatbányászati munka befejeztével számos pozitív következtetést lehet levonni. Az adathalmaz széles körben feldolgozásra került és sikeresen létre jött egy osztályozó modell, aminek a teljesítménye / pontossága kielégítő. A dolgozat részletesen bemutatja egy adatbányászati munkaprojekt általános lépéseit és olyan gyakori hibákat, amik előfordulhatnak egy ilyen fejlesztés során. Szemlélteti azt, hogy a munka során nem feltétlenül lehetséges az, hogy első próbálkozásra kielégítő eredmény születik. Ilyen esetben szükséges neki látni a feladat újbóli értelmezésének és különféle eljárások alkalmazásának.

Az irodalomkutatás segítségével számos hasznos megközelítést lehetett találni, amivel a modellpontosítás megoldható volt. Az irodalom leggyakrabban olyan eljárásokat ajánl, amivel az adathalmazt bővíteni tudjuk mind tulajdonság szempontjából és méret szempontjából. A bővített adathalmaz részletesebb feldolgozásával már nagyobb rálátás volt a mintákra, viszont az adathalmazban további jelentős problémák jelentek meg, amiknek a kiküszöbölése elengedetlen volt.

Ilyen kritikus probléma volt az adathalmaz mérete és a kiegyensúlyozatlansága. Ezeknek köszönhetően a gépi tanuló algoritmusok képtelenek voltak valós és jó teljesítményt nyújtani, mivel túlillesztettek és rátanultak olyan eredményekre, amik több mint valószínű, hogy hamisak. Így minden gépi tanuló algoritmus körülbelül 90-95 százalékos pontosságot eredményezett, ami egy nagy ugrás volt a 60 és 65 közötti százalékos pontosságról. Ahhoz, hogy ezt kiküszöböljük, alkalmaztunk különféle mintavételezési eljárást és keresztvalidációs eljárást. Ezáltal a modellépítés során megbízhatóbb eredményeket lehetett kapni, aminek a pontossága 72-77 százalék körül mozgott. Többféle gépi tanuló eljárás alkalmazásával bebizonyosult, hogy a vizsgált adathalmazra a legalkalmasabban a Support Vector Machine osztályozó és a Naive Bayesian osztályozási eljárás alkalmazható. Ezek közül pedig a Naive Bayesian volt az, amelyik a legpontosabb modellt képes volt létrehozni 76.3 százalékos pontossággal.

Az adatbányászati munka befejezte után készült a dolgozathoz egy webalkalmazás is, aminek a segítségével betekintést kaphatunk a munka feltáró elemzésének és az osztályozási modellek túlnyomó többségének a részébe. Az alkalmazás egy vizualizációs webapplikációnak felel meg, ahol részletesebben beletekinthetünk a verseny eredményeibe, éves és részletes napi bontásokra rendezve. Az osztályozási modell az

alkalmazáson belül használható, ami egy előrejelzést tud adni a versenyző teljesítményéről a felhasználó adatai alapján.

9. Továbbfejlesztési lehetőségek

Egy adatbányászati projekt fejlesztése a legtöbb esetben egy hosszas folyamat. Számos iterációt végig lehet vezetni a munka során és ezt egészen addig lehet folytatni, amíg el nem ér a modell egy kívánt pontosságot. Ebből az következtethető, hogy mindig lehet tovább fejleszteni a teljesítőképességét a gépi tanuló algoritmusnak. A munka továbbfejlesztésénél számos célpontot lehet még megfogalmazni

9.1. Hiperparaméter hangolás

Az irodalomkutatás alapján, ezt alkalmazva további teljesítmény növekedést lehet elérni a gépi tanuló algoritmusoknál, viszont a dolgozatban már nem maradt rá elegendő idő ezeknek az implementálására. Ezen eljárás alatt azt értjük, amikor az optimális hiperparaméterek készletét választjuk ki egy tanulási algoritmushoz. Ezek olyan argumentumok halmaza, amelyeknek az értéke a tanulási folyamat megkezdése előtt kerülnek beállításra az alkalmazáson belül. Alkalmazásuk az algoritmusokon nagyobb teljesítményt eredményezhet.

9.2. Új célok megfogalmazása

A dolgozat elkészítését kezdetlegesen egy fő cél motiválta, az pedig az volt, egy olyan osztályozó modell készüljön, ami a lehető legpontosabban képes megjósolni egy jövőbeli verseny kimenetelét egy versenyző számára. Emellett az is cél volt, hogy egy olyan kimutatást készítsünk, ami jól és látványosan prezentálja a verseny eredményeit különböző táblázatok és diagramok segítségével.

Ezeken a célokon túl számos más célokat is érdemes lehet figyelembe venni, mint például egy futónak a teljesítőképességét a versenyen. Ehhez egy olyan prediktív modell fejlesztése lenne a cél, amely képes a lehető legpontosabban előre jelezni a verseny végeredményét egy versenyző számára percben. Először is megkellene határozni, a hibahatárt és azt, hogy mekkora legyen ennek a mértéke. Ilyen témában lehet találni hasonló kutatást részletes dokumentációval, így ezeket alkalmazni lehetne a Balaton Szupermaraton egyéni eredményeinek a halmazán is.

Amennyiben a kimutatásokon szeretnénk javítani, alkalmazhatóak különféle, modernebb vizualizációs könyvtárak, amiknek a segítségével többféle típusú diagramon lehetne ábrázolni az elérhető adatokat. Mivel több országból és városból is részt vettek a versenyen a versenyzők, ezért egy olyan térképet is meglehetne valósítani, amin jól prezentálhatóak a versenyzők országai és városai. Ehhez különféle geolokációs könyvtárakra van szükség, amiknek az alkalmazásával elkérhetőek országkód és város alapján a koordináta adatok.

9.3. További együttműködés a szervezőkkel

Mivel ennek a versenynek van egy olyan közössége, amely igényt tartana hasonló kimutatásokra, mindenképpen érdemes lehetne folytatni a szervezőkkel a további közös munkát. Mivel a munka során az adatpótlás miatt fel kellett keresnem az illetékeseket ezügyben, akik nagy érdeklődéssel fogadták a témát.

Együttműködve a kimutatásokat lehet pontosítani és értékelni. A közös munka eredményét természetesen nyilvánossá lehet tenni, hogy a versenyen résztvevők és használni tudják. Amint említettem a nagy és aktív közösségnek köszönhetően az igény megvan rá, már csak az alkalmazásra várnak, amely teljesíti a követelményeiket.

9.4. Alkalmazás továbbfejlesztése

Mivel az alkalmazás jelenleg egy egyserű streamlit alkalmazásként fut, ezért ennek a bővítése, személyre szabása nem a legegyszerűbb, néhány esetben lehetetlennek tűnő feladat. A legoptimálisabb megoldás az lenne, ha egy olyan keretrendszerrel kerülne fejlesztésre a webalkalmazás, ami jobban személyre szabható. Ehhez olyan fejlett webes technológiák alkalmazására lenne szükség, amelyek képesek kielégíteni a modern webalkalmazások követelményeit, mind gyorsaságban, minőségben és személyre szabhatóságban. Ilyen fejlett webtechnológia például a React, az Angular vagy esetleg a Vue, amelyek mind támogatják a JavaScript és TypeScript nyelveket, amelyek szintén nagy népszerűségnek örvendenek.

Amennyiben viszont kielégítő az alkalmazás működése és megjelenése, ennek a továbbfejlesztésén és szépítésén is lehet gondolkodni közösen a szervezőkkel. A véleményüket kikérve olyan adatokat is meglehetne jeleníteni a weboldalon, amiket a

területi tudások alapján még fontosnak tartanak és ami a futóközösség számára is kifejezetten érdekes lehet.

Irodalomjegyzék

[1] Gyarmati Péter, Gondolatok a mesterséges intelligencia, a gépi tanulás kapcsán, 2019.

[2] "Half marathon finish time prediction," [Online]. Available: https://towardsdatascience.com/half-marathon-finish-time-prediction-part-1-5807760033eb [Hozzáférés dátuma: 09-05-2022]

[3] "Childhood Ultra-Marathon Runner Study," [Online]. Available: https://www.kaggle.com/code/aiaiaidavid/childhood-ultra-marathon-runner-study [Hozzáférés dátuma: 03-05-2022].

[4] "Decision Tree Classifier" [Online]. Available: https://www.sciencedirect.com/topics/computer-science/decision-tree-classifier#:~:text=The%20decision%20tree%20classifier%20(Pang,possible%20values%20for%20that%20attribute. [Hozzáférés dátuma: 01-05-2022].

[5] "Random Forest Pros and Cons" [Online]. Available: https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04 [Hozzáférés dátuma: 02-05-2022].

[6] "Machine Learning" [Online]. Available: https://www.oreilly.com/library/view/machine-learning-with/9781787121515/697c4c5f-1109-4058-8938-d01482389ce3.xhtml [Hozzáférés dátuma: 03-05-2022]

[7] V. Gudivada, A. Apon, and J. Ding. Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations". In: International Journal on Advances in Software 10.1 (2017), pp. 1 - 20.

[8] "Dealing with very small datasets" [Online]. Available: https://www.kaggle.com/code/rafjaa/dealing-with-very-small-datasets/notebook [Hozzáférés dátuma: 04-05-2022]

[9] Foster Provost, Machine Learning from Imbalanced Data Sets 101, : *AAAI Technical Report WS-00-05*, 2000.

[10] Ozgur Demir-Kavuk, Mayumi Kamada, Tatsuya Akutsu and Ernst-Walter Knapp, Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features, Demir-Kavuk et al. BMC Bioinformatics 2011, 12:412

[11] Ying Zhang and Chen Ling, A strategy to apply machine learning to small datasets in materials science, *Shanghai Institute of Ceramics of the Chinese Academy of Sciences*, 2018.

[12] Thair Nu Phyu, Survey of Classification Techniques in Data Mining, *International MultiConference of Engineers and Computer Scientists 2009 Vol I IMECS 2009, March 18 - 20, 2009*

[13] Foster Provost, Victor S. Sheng, Panagiotis G. Ipeirotis, Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers*, 2008*

[17] Dimitris Kanellopoulos, Handling imbalanced datasets: A review*,* GESTS International Transactions on Computer Science and Engineering, Vol.30, 2006

[18] V. Gudivada, A. Apon, and J. Ding. Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations". In: International Journal on Advances in Software 10.1 (2017), pp. 1 - 20.

[19] Senele Zwelisha Sithole, Data Mining for Imbalanced Datasets: An Overview*, 2005*

[20] Seyda Ertekin, Jian Huang, Leon Bottou, C. Lee Giles, Learning on the Border: Active Learning in Imbalanced Data Classification*, 2007*

[21] V. Ilango, R. Subramanian, V. Vasudevan, A Five Step Procedure for Outlier Analysis in Data Mining*,* European Journal of Scientific Research ISSN 1450-216X Vol.75 No.3 (2012), pp. 327-339

[22] Knechtle, Beat ; Nikolaidis, Pantelis Theodoros, The age of the best ultramarathon performance – the case of the “Comrades Marathon” *Zurich Open Repository and Archive, University of Zurich*, 2017.

Mellékletek

Mappaszerkezet

```
+marathon_statistics
+---dataset
|   +---csv
|   |   +---bszm_2008_2020
|   |   |   bszm.csv
|   |   +---cleaned
|   |   |   bszm_cleaned.csv
|   \---xlsx
|           bszmEredmenyek.xlsx
+---documentation
|   +---prezi
|   |   |   csizmazia_mate_xi32is_apr25
|   +---signed
|   |   2022_2_CsizmaziaMate_XI32IS_Diplomadolgozat.docx
+---notebook
|   |   calculate_length_of_sections.py
|   |   ultramarathon_individual_results.ipynb
|   |   ultramarathon_eda.ipynb
|   |   ultramarathon_classification.ipynb
|   |   ultramarathon_svm.ipynb
|   |   ultramarathon_neural.ipynb
|   |   ultramarathon_bayesian.ipynb
+---publications
|   |   publication_list
+---static
|   +---images
|   |       running.png
|   marathon_statistics_app.py
|   README.md
|   requirements.txt
```

Ábrajegyzék

1. ábra: Data Science Venn diagram 12

2. ábra Döntési fa 28

3. ábra DataPrep riport 34

4. ábra DataPrep korreláció kiértékelés 35

5. ábra Felhő platform felépítése 43

7. ábra Statisztikai felület 2 44

8. ábra Adathalmaz feltöltése felület 45

Táblázatjegyzék

1. táblázat kiinduló adathalmaz tulajdonságai típusmegjelöléssel 24
2. táblázat bővített adathalmaz tulajdonságai típusmegjelöléssel 31
3. táblázat Korcsoportok 32
4. táblázat K-fold Cross Validation minta 36
5. táblázat Stratified K-fold Cross Validation minta 37
6. táblázat gépi tanuló algoritmusok összehasonlítása 40