Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Szalóki Kristóf

TDK dolgozat

GÉPI TANULÁS ZAJOS CÍMKÉVEL RENDELKEZŐ ADATHALMAZBÓL

KONZULENS

Dr. Szűcs Gábor

BUDAPEST, 2018

Tartalomjegyzék

1 Bevezetés

Az utóbbi időben egyre szélesebb körben terjedtek el a mesterséges intelligencián alapuló döntéstámogató rendszerek. A rendszerek kielégítő működésének az alapja a megfelelő mennyiségű és minőségű adatok rendelkezésre állása. A feladatokhoz felhasznált gépi tanuló modelleket azonban sok esetben nem készítik fel a valós életből származó pontatlan információinak megfelelő feldolgozására. A dolgozat célja egy olyan osztályozási megoldás kidolgozása volt, mely az adatforrásban hibásan szereplő, úgy nevezett zajos adatok által keltett hibákat minimalizálni tudja még olyan esetekben is, amikor a tanuló halmazban nem kiegyensúlyozott az egyes osztályok aránya. A módszer segítségével kiszűrhetők azok az adatok, melyek negatív irányba befolyásolják a gépi tanuló algoritmus működését. A megoldás során különféle zajszűrési technikákat alkalmaztam, melyek segítségével méréseket készítettem a legmegfelelőbb irány meghatározása érdekében. A munka végén az elkészített modellt teszteltem kiegyensúlyozatlan adathalmazokon végzett teljesítmény szempontjából is. A dolgozatomban felvetett probléma egyre nagyobb jelentőségű lesz, ahogy egyre nagyobb hatáskört biztosítunk a mesterséges intelligenciának, hiszen nem lenne szerencsés, ha kritikus döntéseket meghozó modellek zajos adathalmazon a helyes döntés helyett rosszul tanulnák meg az összefüggéseket.

2 Gépi tanulás zajos adathalmazokon

2.1 Zajos adathalmazok és létrejöttük oka

Mindennapi cselekvéseink, nagyrésze meggondolt döntéseken alapszanak. Még abban az esetben is, ha a cselekvő a tudatosságból semmit sem észlel. Ahhoz, hogy ezek a döntések a lehető legjobbak legyenek rendkívül sok információra van szükség. A környezetünkből vett információk során tehát nyugodtan élhetünk azzal a előfeltételezéssel, hogy a mintavételezett információ, melyet később adatokká alakítunk át zajosak lesznek egy bizonyos mértékben. A dolgozat jelen probléma megoldásaival foglalkozik, feltárja a zajok különféle típusait, valamint bemutat két megoldási lehetőséget is a problémakör leküzdésére.

Adatelemzésben zajnak nevezzük a nagyméretű jelentéssel nem rendelkező információk összeségét. Valamint zajnak nevezzük azokat az adatokat is, melyeket a gépek nem tudnak megfelelően vagy helyesen interpretálni.

A korábbi bekezdés alapján valós adathalmazokkal való munka során sokszor kell majd olyan adatokkal dolgozni melyeket zajosnak tekintünk.

$$Adat = Valós\ információ + Zaj$$

Azok az adathalmazok, melyekre szükség van a feladatok megoldására nem mindig állnak a rendelkezésünkre, így bizonyos esetekben azokat nekünk kell létrehozni. Adathalmazok létrejövetele során, alapvetően valamiféle adatgyűjtést végzünk. Ez végezhető emberek, illetve gépek által is, mindkét szereplőnek van létjogosultsága ezen feladatok elvégzésére, szenzor adatok gyűjtésére bőven elég a gépi jelenlét, míg szociális témájú adatok gyűjtésében az emberek sokkal hatékonyabbak. Bár sokszor figyelmen kívül hagyjuk, az adatgyűjtés sok esetben jelentős anyagi ráfordítással járhat, melynek mértéke összefügg az adathalmaz minőségével, illetve tisztaságával. A zajoknak két jelentős forrása lehet, implicit zajnak nevezzük azokat a hibákat, melyek a mérőeszköz vét mintavételezés közben. Illetve véletlen zajnak nevezzük azokat a hibákat, melyeket szakemberek vétenek az adatgyűjtés, illetve feldolgozás során, mint például a gyűjtött adatok digitalizálásával [2].

2.2 Adathalmazon jelentkező zajok típusa

Zajok esetén alapvetően megkülönböztetünk két fő csoportot, melyek a címkézési hiba, valamint az attribútum hiba, ezeken belül természetesen léteznek különféle hibák is:

- **Címkézési hiba**:
  - Ellentmondásos példány
  - Félre címkézett példány
- **Attribútum hiba** (Attribute Noise):
  - Hibás érték (Erroneous value)
  - Hiányzó érték (Missing value)
  - Nem értelmezhető érték (Do not care value)

ID	Attr0	Attr1	Attr2	Cél
1	Anglia	11	2.3	*Európa*
2	Brazília	7	3.4	***Ázsia***
3	Anglia	11	2.3	***Afrika***
4	Kína	3	1.9	*Ázsia*
5	Franciaország	2	2.9	*Európa*

1. táblázat

Címkézési hiba, amikor egy példány helytelenül van felcímkézve, ennek oka számos eshetőségre vezethető vissza: eltérő háttértudással elvégzett címkézés, szubjektivitás címkézés alatt vagy adatbeviteli hiba is lehet a kiváltó ok.

- Ellentmondásos példány: Azon példányokat nevezhetjük ellentmondásos példányoknak, melyekből legalább két azonos előfordulás található az adathalmazban, azonban legalább két különböző osztálycímkével rendelkeznek. Az első táblázatban ez a 1 és 3 ID-val rendelkező sorok.
- Félre címkézett példány: Félreosztályozottnak nevezzük azokat a példányokat, melyek összes előfordulásakor helytelen osztálycímkét kapnak. Az első táblázatban a 2 ID-val rendelkező sor egy félre címkézett

példány. Fontos megjegyezni, abban az esetben, ha a táblázatban szerepelne az alábbi sor (ID:6, Attr0: Brazília, Attr1:7, Attr2:3.4, Cél: Amerika), akkor az egyben ellentmondásos példány is lenne.

ID	Attr0	Attr1	Attr2	Cél
1	Anglia	11	2.3	*Európa*
2	Brazília	**7.34**	3.4	*Amerika*
3	Anglia	11	2.3	*Európa*
4	**?**	3	1.9	*Ázsia*
5	Franciaország	2	**9999**	*Európa*

2. táblázat

Attribútum hiba: Az adathalmaz egy rekordjának legalább egy attribútuma helytelen értéket tartalmaz, akkor attribútum hibásnak nevezhetjük a rekordot.

- Hibás érték: Azt az értéket nevezzük hibás értéknek, mely az attribútum típusától eltér, illetve az adott attribútum értékhalmazán kívül esik. A 2. táblázat második sora szemlélteti ezt az esetet, ahol az Attr1 csak egész értéket vehetne fel.
- Hiányzó érték: Abban az esetben, ha nem ismerjük az attribútum értékét, például a táblázat 4. sorában nem ismert az ország neve, így helyén egy „?” karakter szerepel.
- Nem értelmezhető érték: A nem értelmezhető érték inkább a hiányzó érték esethez áll közelebb mintsem a hibás értékhez. Abban az esetben, ha nem lehetséges az adott érték meghatározása, egy előre egyeztetett szótár alapján választják meg a helyes „nem értelmezhető értéket”, mely még így is hordoz némi információt magában. A táblázat utolsó sorában a Attr2 helyen álló 9999 érték mögött állhat jelentésként, hogy nem kívánt válaszolni, illetve az is hogy nem tudta a választ. Így némi információ többlethez jutunk, azonban az osztályozást önmagában ez nem segíti, mivel maga az eset zajosnak tekinthető.

A dolgozatom további részében a címkézési hibák lesznek a fókuszban, ugyanis a szakirodalom nagyobb része a könnyebben megoldható attribútum hibákkal foglalkozik és viszonylag kevés publikáció született a címkézési hiba témakörben, holott ugyanolyan fontosságú probléma terület ez is. Így zajos adathalmaz alatt a továbbiakban tehát csak a címkézési hibák által előállt adathalmazt értem. A munkám során az attribútum hibáktól eltekintettem, és továbbá a kutatásom fő célja a zajos címkézésű elemek detektálása volt [3].

2.3 Indikátorok címkézési hibákkal terhelt adathalmazok számára

Az adatokban rejlő információ kinyerése sokkal könnyebb tiszta adatokon, mintsem zajokkal szennyezett adatokon. Egy lehetséges cél olyan modell építése, amely a lehető legkisebb mértékben érzékeny a zajos címkékre, ezt a tulajdonságot robosztusságnak nevezzük. Egy bizonyos modellt abban az esetben nevezünk robosztusabbnak, ha az jobban teljesít zajos adathalmazon, mint az összehasonlításban szereplő másik modell [2]. A modell jóságára a következőkben ismertetett mérőszámok állnak rendelkezésre.

Pontosság:

A pontosság (Accuracy) egy elég fontos mérőszám, mint ahogy az alábbi képlet is szemlélteti, minden csoportot figyelembe veszünk ez érték meghatározásakor.

$$Pontosság = \frac{TP + TN}{TP + FP + FN + TN}$$

TP= Valós pozitív esetek száma

TN= Valós negatív esetek száma

FP= Hamis pozitív: olyan negatív esetek száma, melyek a pozitívak közé lettek sorolva tévesen

FN= Hamis negatív: olyan pozitív esetek száma, melyek a negatívak közé lettek sorolva tévesen

A pontosság által kapott eredményünket azonban a kiegyensúlyozatlan adathalmazok esetén feltételekkel kell kezelni. Tegyük fel, hogy egy adathalmazon bináris osztályozást szeretnénk végezni és az A osztálybéli elemek a teljes halmaz 99% át teszik ki. Ebben az esetben, ha az összes elemet automatikusan az A csoportba sorolunk gondolkodás nélkül, akkor is 99%-os pontosságot érünk el [6].

Robosztusság mérése:

Pontosság relatív csökkenése (The **R**elative **L**oss of **A**ccuracy):

$$RLA_{x\%} = \frac{A_{0\%} - A_{x\%}}{A_{0\%}}$$

A= Accuracy pontosság mértéke [0,1]

X= A zajos adatok aránya [0,1]

A módszer előnye, hogy könnyedén alkalmazható, valamint annak a modellnek melynek nagyobb lesz a pontossága a tiszta adathalmazon, kisebb lesz az RLA értéke [5].

Zaj csökkenés mérése:

Egy másik megközelítés mikor a zaj csökkenésének a mértékét vesszük kulcs indikátornak, és ennek segítségével mérjük vissza a modellünk teljesítményét. Iteratív tanítás során az iterációk végén informatív mérőszámként szolgálhat [10].

Zaj csökkenés mértéke (**N**oise **E**limination **P**recision):

$$NEP = \frac{\| K \cap C \|}{\| C \|}$$

K=Azon halmaz egyedei, melyek zajosnak lettek osztályozva

C= A zajos egyedek halmaza

2.4 Zajszűrési módszerek

A zajos adathalmazokon alkalmazni kell bizonyos operációkat annak érdekében, hogy az osztályozó algoritmusunk jobb eredményeket kapjon. A szakirodalom két fő irányból közelíti meg a zaj szűrési módszereket.

Adatelőkészítési: Az adatelőkészítés során különféle szűréseket és egyéb műveleteket hajtunk végre annak érdekében, hogy javítsunk az adathalmaz minőségén.

Algoritmikus: Olyan robosztus osztályozó algoritmusokat keresünk, melyek jobban kezelik a zajt, és annak esetleges növekedése sem jelenti az eredmény drasztikus csökkenését.

Az algoritmikus módszerek kevéssé adaptívak, valamint bizonyos esetben az osztályozó algoritmusok működésén is változtatni kell, melynek implementálása akár rendkívül bonyolult is lehet. „Általánosságban az osztályozó algoritmusok pontossága felskálázás esetén romlik az egész adathalmazon végzett osztályozáshoz képest, ez különösen igaz zajos adathalmazok esetén” [7]. Adatelőkészítés során számos szűrési módszert alkalmazhatunk, a megoldás előnye, hogy külön választhatjuk a zaj detektálást, valamint a tanulást; így csökken a zajok általi információ torzítás, valamint a tanuló algoritmus megváltoztatására sem lesz szükség [11].

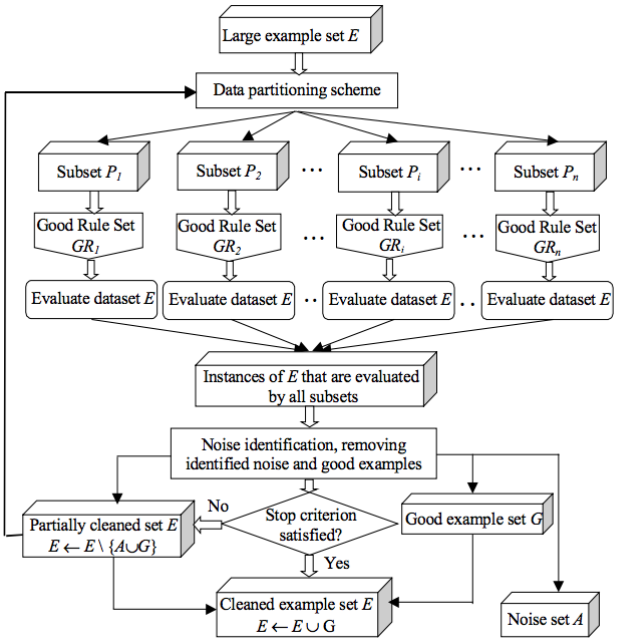
Adatelőkészítés során használt zajszűrő módszerek.

All K-Nearest Neighbour (All kNN): A módszer során a tisztított adathalmaz (TI) elemei közé felvesszük az összes tanító adathalmazba (TA) szereplő példányt, majd TI minden elemét megvizsgálva abban az esetben tartjuk meg ha k legközelebbi szomszédjai egyértelműen meg tudják határozni az osztályát. All kNN esetén k-szor ismételjük meg ezt a folyamatot, mely során a figyelembe vett szomszédok száma megegyezik az iteráció aktuális sorszámával. Abban az esetben, ha már egy iterációnál is félreosztályzás történik abban az esetben jelöljük, hogy a példány zajos, és a későbbiekben eltávolítandó. Az eltávolításra csak a vizsgálat végén kerül sor, azaz az összes zajos egyed egyidőben lesz eltávolítva a TI adathalmazból [8]. Az algoritmus részletes bemutatása a 2.6 fejezetben található.

Metric based filter (MF): Egyszerű szűrő, mely a tanító adatokon való vizsgálatot követően meghatározza az adott elem zajos valószínűségét. Azon elemek, melyek valószínűsége meghaladja

ezt a küszöböt eltávolításra kerülnek az adathalmazból, és végül előáll a tisztított tanuló adathalmaz [11].

Partitioning filter (PF): Egy meghatározott küszöbérték eléréséig iteratív végrehajtás során a zajosnak jelzett adatokat eltávolítjuk az adathalmazból. Az iterációs folyamat egészen addig tart, míg egymást követő iterációk során a zajosnak ítélt elemek száma kevesebb nem lesz, mint az azt megelőző iteráció során zajosnak talált egyedek egy paraméterben meghatározott százaléka. Az algoritmus első lépéseként a tanító adathalmazt (E) n darab egyenlő méretű részhalmazra bontjuk fel. Egy kiválasztott algoritmus segítségével osztályozó modellt építünk a részhalmazokon, majd pedig kiértékeljük az egész tanító adathalmazon. Ezt követően szavazásos alapon (mely lehet konszenzus, vagy többségi alapú), azon elemeket, melyeket zajosnak tekintettünk eltávolítjuk az E halmazból és egy Zajos halmazba (A) helyezzük. Abban az esetben, ha az adott elem minden osztályozó szerint helyesnek tekinthető a Helyes halmazba (G) kerül. A megmaradt, részben tisztított adathalmazon tehát tovább iterálunk egészen addig amig a kilépési határérték alá nem megyünk a zajos arány tekintetében [9]. A partitioning filter jelentős szerepet kapott a munkám során. Az általam készített megoldás alapját is ez a modell szolgáltatta, a kiterjesztett modellem részletes bemutatására a 3. fejezetben kerül sor.



1. ábra A partitioning filter működése forrás:[9]

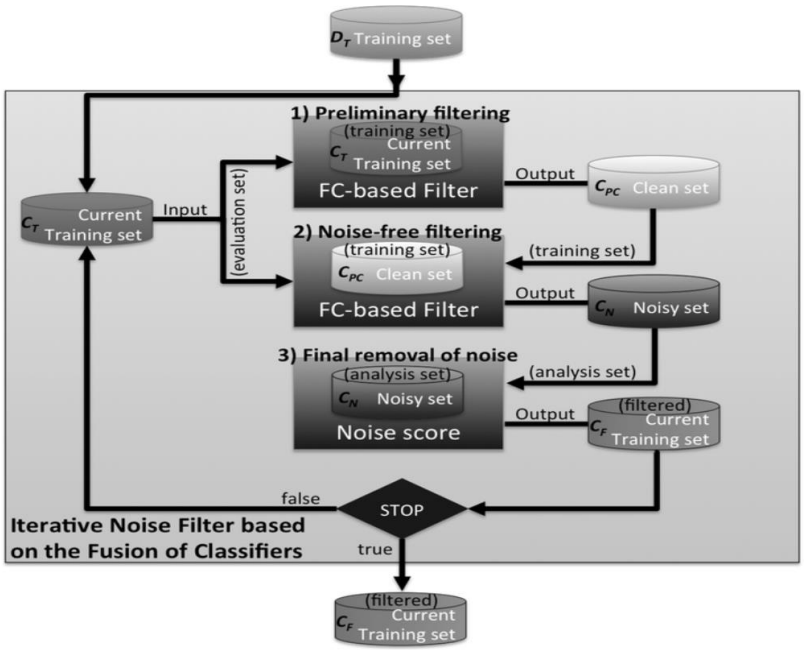
Ensembles filters: A tanuló adathalmaz részhalmazain több osztályozó modell alkalmazásával próbálja megtalálni a zajos egyedeket az adathalmazon. A zajos egyedek megtalálása érdekében egy K-keresztvalidációt alkalmaznak a tanuló adathalmazon N darab osztályozó segítségével. Az algoritmus első lépésként K darab részhalmazra osztja fel a tanító halmazt. Az összes N darab osztályozó esetén az összes K részhalmazra az osztályozó algoritmust a K-1 részhalmazon tanítjuk, majd a maradék részhalmazon kiértékeljük az összes osztályozót. Eredményül minden egyes osztályozó felcímkézi az összes példányt, majd szavazással határozzák meg, hogy melyek helyes esetek, és melyek eltávolítandó zajos példányok [12].

Iterative class noise filter based on Fusion of Classifiers (INFFC): Több féle osztályozó, és zajszűrési technika alkalmazásával iteratív módon távolítja el a zajos egyedeket az adathalmazból. A módszer a három fő zajszűrési módszeren alapszik, melyek a:

- Metric based filter
- Ensemble filters

- Partitioning filter

Minden iteráció az első lépésében eltávolítja a zajos példányok egy részét, hogy azok a későbbi lépéseket ne befolyásolják. Ezt követően egy új szűrés következik, mely a részben tiszta adathalmazt használja tanulásra, majd a teljes tanuló adathalmazon alkalmazzuk. Ezt követően elő is áll a tiszta adathalmaz, valamint a zajos adatokat tartalmazó adathalmaz.



2. ábra Az INFFC modell működése forrás: [11]

A szakirodalomban sokszor konzervatív modellként hivatkoznak az INFFC-re, mivel a korábban említett szűrő modellekhez képest ez szűri ki a legkevesebb tiszta adatot [13].

2.5 Kiegyensúlyozatlan adathalmazok kezelése

Legjobban kezelhetők azok az adathalmazok, ahol osztályok eloszlása egyenletes, azonban előfordulnak olyan esetek, amikor az egyik típusú osztály tagjai jelentős többségben vannak a másikakhoz képest. Ilyenkor kiegyensúlyozatlan adathalmazról beszélünk. Bizonyos esetekben ez extrém mértéket is ölthet, vegyük például a banki csalásokat, amik szerencsére ritkán fordulnak elő. Ez azonban azt is jelenti, hogy a témával foglalkozó adathalmazban a csalást reprezentáló osztály számossága kevesebb, mint 1%. Gépi tanulás során azonban vannak olyan algoritmusok,

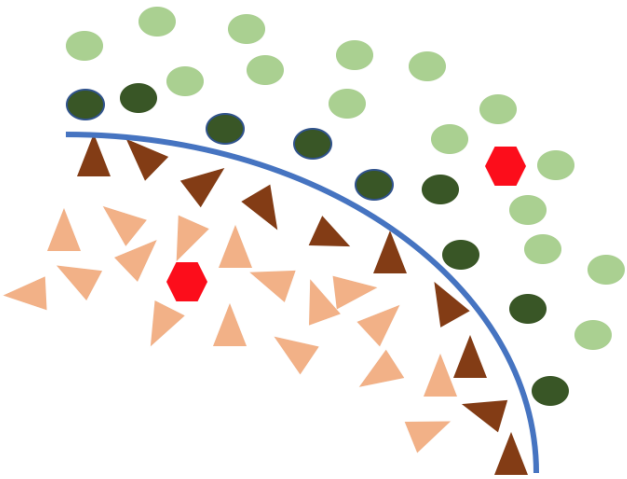
melyek érzékenyek erre a kiegyensúlyozatlanságra, keresni kell tehát olyan algoritmusokat, melyek jól teljesítenek szélsőséges körülmények között is. Kutatók rámutattak, hogy az osztályok aránytalansága önmagában nem a legnagyobb probléma, az osztályozási képességet más befolyásoló tényezők, mint például a zajos adatok, és a szélsőséges elemek eloszlása tudja jelentősen csökkenteni.

Az adathalmazban háromféle kategóriába csoportosíthatjuk a példányainkat:

Biztonságos, tiszta példányok (Safe): Olyan helyes példányok tartoznak ebbe a csoportba, melyek osztályba való tartozása egyértelműen meghatározható.

Zajos példányok (Noisy): Olyan zajos példányok, melyek a másik osztály elemei közé van beágyazva.

Szélsőséges példányok (Borderline): Olyan példányok, melyek több osztály példányaival vannak szorosan körülvéve, és a megfelelő osztályba való tartozásuk közel sem magától értetődő.



3. ábra Szélsőséges példányok és zajok viszonya

A zajos és szélsőséges csoportba tartozó elemek eltávolításával javítani lehet az osztályozó performanciáját, erre többféle lehetőség van: SMOTE [15] + Szűrő algoritmus [14].

Kiegyensúlyozatlan adathalmazok kezelésére három lehetőség van:

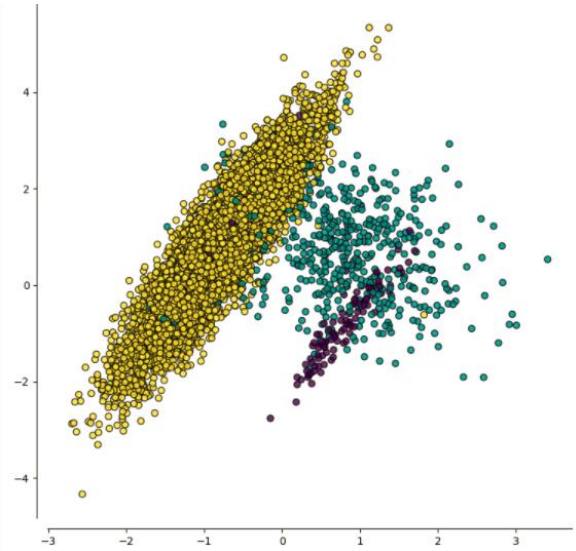
- Megőrizzük az eredeti adathalmazt, és speciális algoritmust választunk hozzá. Ebben az esetben közelítünk legjobban a valósághoz, mivel nem végzünk módosítást. Az adathalmaz karakterisztikája viszont nagyban redukálja a lehetséges algoritmusokat.

- Alul-mintavételezzük az eredeti halmazt annak érdekében, hogy közel megegyező arányú legyen az osztályok aránya. A megoldás során a kisebbségben lévő értékeket megtartjuk teljes egészében, a másik adathalmazból pedig megfelelő mennyiségű adatot választunk véletlenszerűen. Ebben az esetben a teljes reprezentáció nem valósul meg, viszont a tanuló algoritmus jobban teljesít, mivel az adathalmazban az osztályok aránya kiegyenlített.

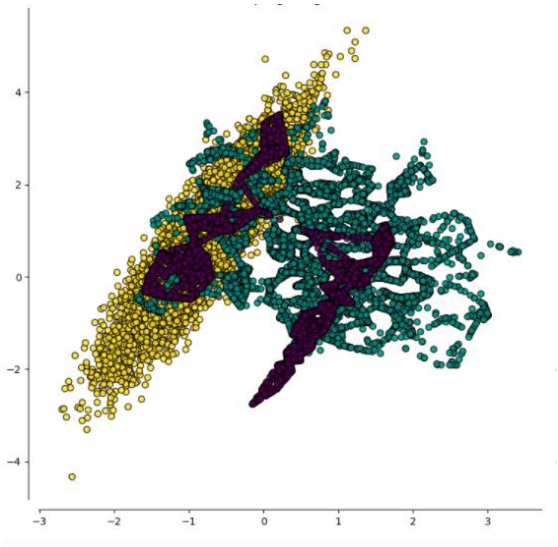
- Túlmintavételezést végzünk az eredeti adathalmazon és a SMOTE (**S**ynthetic **M**inority **O**versampling **Te**chnique) [15] módszer segítségével új értékeket állítunk elő a kisebbségben levő osztály elemeiből. A SMOTE módszer szerint venni kell egy lemet a kisebbségi halmazból (X_i) valamint ennek a k legközelebbi szomszédját. Ha k=3, akkor ezek legyenek x_z_0, x_z_1, x_z_2, majd ezt követően véletlenszerűen választunk egy elemet közülük. Az interpolációval előállított X_N érték pedig a kiválasztott x_z_i és X-et összekötő egyenesen található [15].

$$X\_N = X_i + \vartheta * (X\_Z_i - X_i)$$

$$\vartheta=[0,1]$$



4. ábra: Az eredeti adathalmaz [15]



5. ábra: Újra mintavételezett adathalmaz (SMOTE) [15]

A 4. ábrán látható három különböző osztály elemei, és azok arányai. Az eredeti adathalmaz erősen kiegyensúlyozatlan a sok sárga osztályú egyed következtében, a SMOTE algoritmus segítségével újra mintavételezett adathalmazban, mely a 5. ábrán látható az osztályok aránya sokkal kiegyenlítetteb, mint az korábban volt. A lila osztályú elemek a 4. ábrán is megtalálhatók azonban a sárga szín elfedi azokat. Az újra mintavételezett adathalmazban már az eredeti, és az újonnan létrehozott lila értékek kerülnek a fedési sorrend tetejére.

2.6 All kNN

Az All kNN zajszűrő modellt választottam ki, mint egyszerű referencia modellt. A modell alkalmazása során egyszer kell csupán végig iterálnunk az egész adathalmazon. A modell működését nagyban befolyásolja a korábban megválasztott K érték. A K érték határozza meg, hogy az elem zajosságát mekkora környezetére alapozva határozzuk meg. Az algoritmus három részfázisra osztható fel, melyek:

- Előfeldolgozás
- K legközelebbi szomszéd vizsgálata
- Kiértékelés

Az algoritmus során használt jelentősebb változók, a K változóval konfigurálható a modell:

Alap adathalmaz: DS

Aktuális központi elem: N_i

Maximálisan vizsgált szomszédok száma: K

Aktuálisan vizsgált szomszédok száma: ε

Zajokat tartalmazó adathalmaz: F

Tiszta adatokat tartalmazó adathalmaz: C

Előfeldolgozás:

Az előfeldolgozás során az adathalmazban található változókat transzformációkkal átalakítjuk a gépi tanuló algoritmusok számára megfelelő formátumra. A kategorikus változókat numerikus változókká transzformáljuk át a megfelelő függvény alkalmazásával. Mivel az algoritmus későbbi fázisában klaszterezést szeretnénk végrehajtani rajta, így az adathalmaz attribútumait normalizálni kell. Normalizálás nélkül a klaszterezés során nem lehetne helyes távolságot számolni és így az eredményt egy-egy kisebb tartományban mozgó attribútum pozitív, míg a széles tartományba mozgó attribútumok negatív irányba befolyásolhatnák.

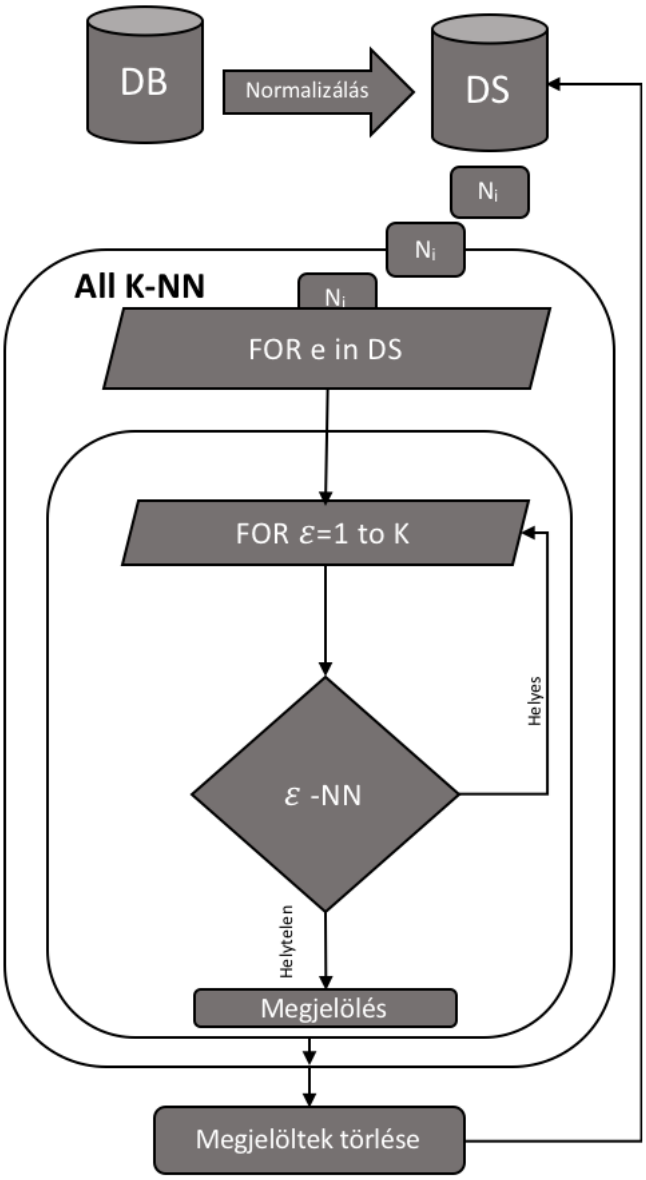
K legközelebbi szomszéd vizsgálat.

Az előfeldolgozást követően az adathalmaz készen áll a kNN algoritmus futtatására. Az algoritmus során az adathalmaz minden egyes elemét sorban vesszük, legyen az aktuális elem N_i. A K paraméter értékétől függő iterációt hajtunk végre, melyek során a középső elemként az N_i elemet tekintjük. Az iterációk során minden esetben vesszük az adott elemnek az ε legközelebbi szomszédját. Az N_i elem szomszédjait az NN (Nearest Neighbour) algoritmus segítségével tudjuk meghatározni. Abban az esetben, ha az elem összes szomszédja vele megegyező osztályba tartozik, abban az esetben adott ε értékre helyes a modellünk és ε értékét növeljük egyel egészen addig amíg ε meg nem egyezik K értékével. Ha ebben az esetben is helyes minden érték, akkor az N_i elemet tisztának tekintjük. Azonban, ha az iteráció során bármely esetben az elem bármelyik szomszédja eltérő attól, akkor kilépünk a ciklusból és zajosnak jelöljük meg az adott elemet. Az N_i eltávolítását nem azonnal hajtjuk végre, hanem a vizsgálatok lezárultát követően, ennek oka, hogy a zajos elemek segíthetnek detektálni a további zajos elemeket az adathalmazban.

Kiértékelés

A zajos elemek kiszűrését követően az adathalmazban már csupán az algoritmus szerint helyesnek ítélt elemek tartoznak. A tiszta adatokat elhelyezzük a C

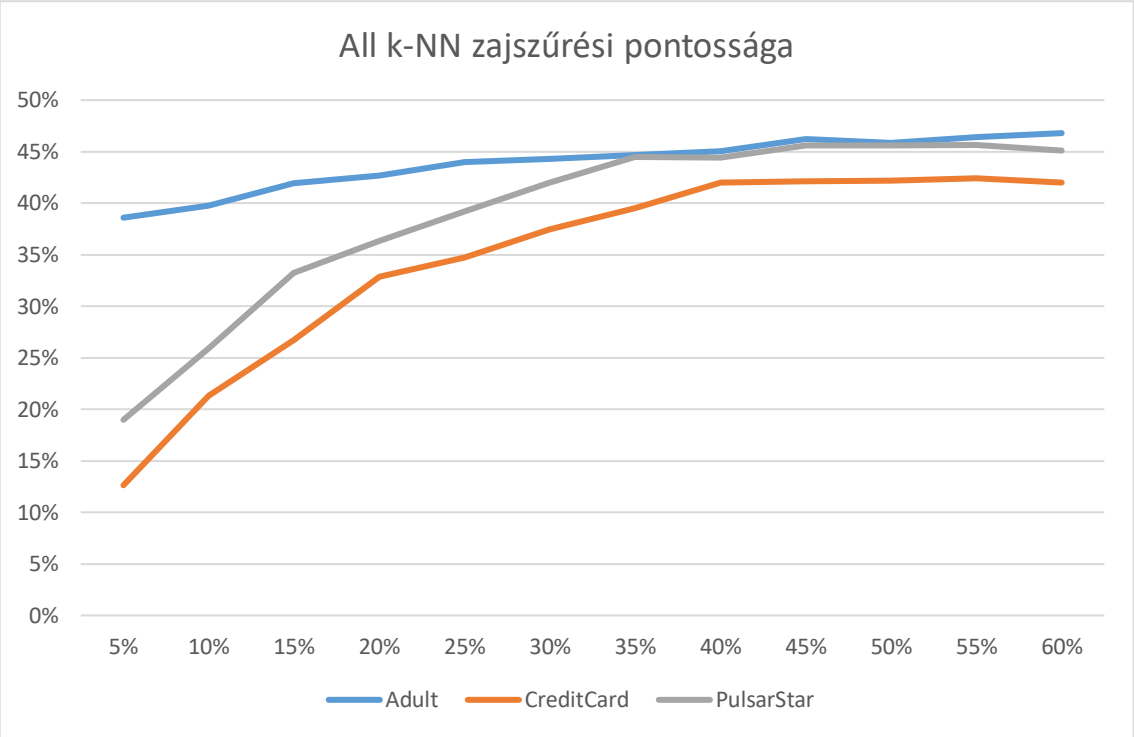
adathalmazban, a továbbiakban ezt fogjuk használni, a zajokat az F halmazba tároljuk el. Az algoritmus előnyei közé tartozik, hogy egyszerű, könnyen implementálható, valamint konfigurálható, csupán a K érték helyes megválasztása szükséges hozzá. Hátránya azonban, hogy nem csupán a zajokat szűri ki, hanem a zajok környezetében található helyes elemeket is.



6. ábra Az All kNN modell működése

Zajszűrési eredmények

Az All kNN algoritmus futtatása során 5-60 %-os zajosságú adathalmazokon teszteltem, működése során látható, hogy a zajok egy részét képes helyesen eliminálni az adathalmazból azonban sok helyes elem is kiszűrésre kerül az algoritmus futtatása során.



7. ábra Az algoritmus zajszűrési pontossága

A 7. ábrán látható, hogy az All kNN modell egyre nagyobb mennyiségű zaj arány esetén jobb teljesítmény nyújt, de még így is legjobb esetben is csupán a zajok 45%- át tudja eliminálni. Az ábra x tengelyén az adathalmazban található zajok aránya szerepel, míg az y tengely a modell által helyesen megtalált és eliminált zajok arányát jelzi.

3 Tervezett zajszűrő modell partitioning filter alapon

Gépi tanuló algoritmusok alkalmazásainak elterjedésével, újabbnál újabb kihívások jelennek meg. A zaj, mint gépi tanulásnál tapasztalható jelenség számos formában előfordulhat, és negatív hatást gyakorolva elrontja a helyesen működő algoritmusaink eredményeit. A zajok kiszűrésére az irodalomkutatásban bemutattam néhány lehetséges megoldási modellt. A munkám során kiválasztottam közülük kettőt, melyek működését mélyebbre hatóan tanulmányoztam. Az *All kNN* modell az egyszerűbb zajszűrő modellek körébe tartozik, implementálása egyszerű, azonban az algoritmus típusából adódóan magas futási idővel kell számolni egy zajszűrési művelet során. A partitioning filter az összetettebb zajszűrő algoritmusként tartják számon, a munkám során elkészített zajszűrő modell alapjául az „Eliminating Class Noise in Large Datasets" tanulmányában bemutatott partitioning filtert vettem [9]. Az alapkoncepció feldolgozását és implementálását követően módosításokat javasoltam és hajtottam végre a jobb zaj szűrési eredmények elérése érdekében. A továbbiakban részletesen bemutatom az általam elkészített **E**xtended **P**artitioning **F**ilter (EPF) működését, majd kitérek a kiegyenlített adathalmazok esetén szükséges intézkedésekre, annak érdekében, hogy a modellek teljesítményére a lehető legkisebb mértékben hassanak az adathalmaz elemeinek jelentősen eltérő arányai. A partitioning filter nem volt felkészítve a kiegyensúlyozatlan adathalmazok kezelésére, valamint a belső működése során bizonyos döntéseket eltérő algoritmus alapján hozott meg, ezért úgy döntöttem, hogy a partitioning filter modellt alapul véve, a szabálykiválasztás valamint a kiértékelésnél is módosítottam annak működési algoritmusát és új logikát vezettem be a modellbe.

Extended partitioning filter

Az EPF zajszűrő rendszer megoldása több részegységből tevődik össze. A modell koncepciója a részekre bontás és az entrópia adta lehetőségeket igyekszik kihasználni. A zajszűrő rendszer célja, hogy iteratív működése során a lehető legkevesebb helyes elem eliminálása mellett az összes zajt kiszűrje az adathalmazból. Az iteráció egészen addig tart, míg az iteráció során detektált zajok aránya egy kritikus szint alá nem csökken. A modell részegységei a következők:

- Partícionálás
- Szabály generálása

- Jó szabály kinyerése
- Szabályok alkalmazása

- Szűrés
- Kiértékelés.

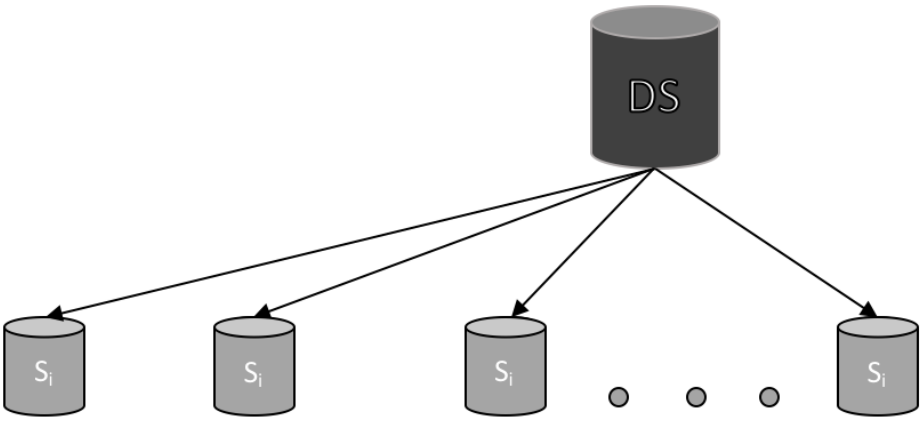
+ Kiegyensúlyozatlanság kezelése

Az algoritmus során használt jelentősebb változók, a $*$ jelölt elemekkel konfigurálható a modell:

Alap adathalmaz: DS
Aktuális tanuló adathalmaz: S_i
Zajokat tartalmazó adathalmaz: F
Tiszta adatokat tartalmazó adathalmaz: C
Részhalmazok száma*: K
Szabály: R
Jó szabályok száma*: N
Helyes adatok visszatevési aránya*: PB
Helyes elemek felső határa hibák esetén*: E_{LTH}

3.1 Partícionálás

Az első részegység a partícionálás, mely során a rendelkezésre álló adathalmaz elemeit részhalmazokra osztjuk szét. A DS halmaz elemeit véletlenszerűen felosztjuk K darab részhalmazra. S_i az aktuális részhalmaz, mellyel a továbbiakban dolgozni fogunk és azt tekintjük tanuló adathalmaznak, míg a $DS \setminus S_i$ elemek a validációs halmaz részei.



8. ábra Partícionálás folyamata

3.2 Szabály generálás

A szabály generálási és alkalmazási egységben minden egyes S_i részhalmazra elvégezzük az alábbi pontokban szereplő műveletek. A szabályok tartalma különböző részhalmazonként eltérő lesz, azonban globálisan tekintve a szabályok az adathalmaz egy csoportját fogják megfelelően meghatározni

Jó szabályok kinyerése

Az S_i halmaz elemei segítségével egy döntési fát készítettem. A döntési fát iteratív bejárva kinyertem abból a fa egyes csomópontjaira vonatkozó szabályok rendszerét. A szabály R egy olyan objektum, ami tartalmazza melyik attribútum mekkora küszöbértékénél kell nagyobb, illetve kisebb értéket felvenni, valamint a szabály által meghatározott osztály típusát is. A szabályok sokaságából azonban csupán a legerősebb szabályokra volt szükségem. A legerősebb szabály meghatározása során az Egyetértési arányt, valamint a Reprezentativitást (Coverage) vettem figyelembe.

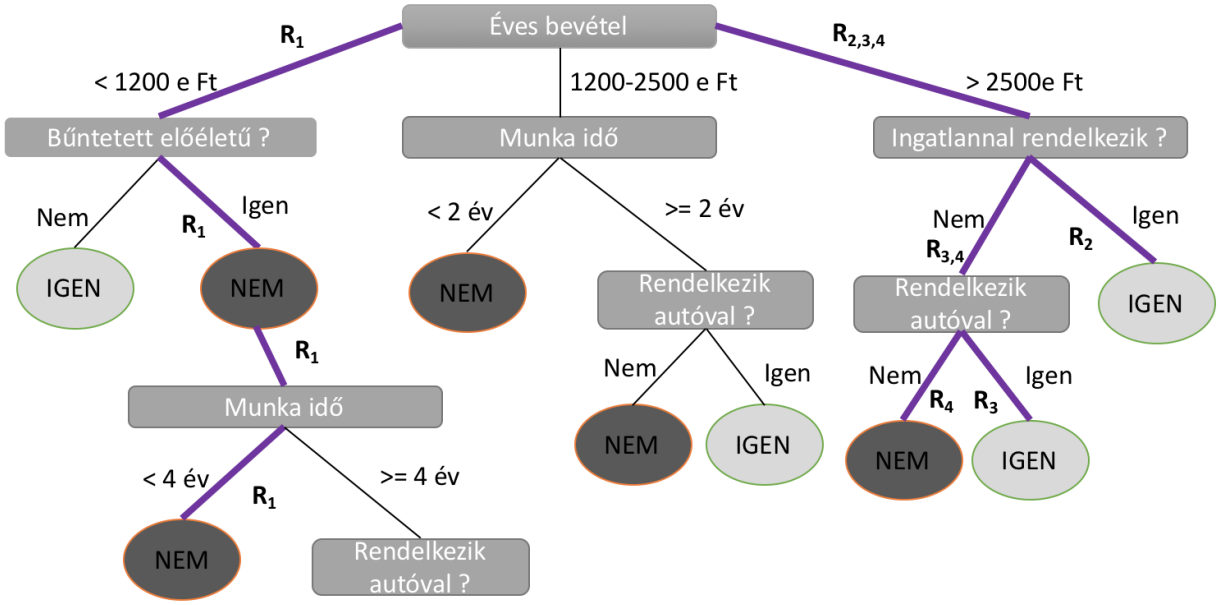
Egyetértési arány: Egy szabály egyetértési aránya a csomópontban található elemek megoszlásából határozható meg. Legyen α az adott csomópontban többségben lévő elemek halmaza, míg β a kisebbségben álló egyedhalmaz.

$$\text{Egyetértési arány} = \frac{|\alpha|}{|\alpha \cup \beta|}$$

Reprezentativitást: Egy szabályt reprezentatívnak nevezünk, ha a csomópontban szereplő elemek száma magas.

$$Reprezentativitás = \frac{|\alpha \cup \beta|}{|S_i|}$$

Külön-külön egyik kritérium sem elegendő a szabály megfelelőségére, azonban együtt már definiálni tudják azt. A szabályokat tehát sorba rendeztem a fent említett kritériumok alapján, majd kiválasztottam belőlük a legjobb N darabot a két metrika együttes alkalmazásával. Az igy előállt halmazt a jó szabályok halmazának nevezem.



9. ábra Döntési fa szabályainak kinyerése

Az ábrán egy döntési fa látható, melyből a kiválasztott csomópontok és élek alkotnak egy szabályt. A döntési fákban számos szabály található meg, azonban a szabályok jósága nagyban különbözik. A fenti ábrán lilával jelölt élek mutatják a négy legerősebb szabályt mely a döntési fában megtalálható.

Szabály R1					
Nagyobb	**$Node_3$**:4,34	**$Node_2$**:1,7	**$Node_6$**:5,4	**$Node_{13}$**:4,3	**$Node_{18}$**:0,3
Kissebb	**$Node_1$**:1,34	**$Node_4$**:2,1	**$Node_9$**:3,3	**$Node_{23}$**:1,2	**$Node_{25}$**:0,4
Egyetértési arány	0,95	**Fedés**	0,32	**Osztály**	1

10. ábra Az R1 szabály felépítése

A 10 ábrán egy szabály található, mely első sora tartalmazza azon csomópontokat, melyben szereplő értékeknél nagyobbnak kell lennie, míg a második sor azokat a csomópontokat, melynél kisebbnek kell lennie az aktuális elem attribútumainak.

Szabályok alkalmazása:

Amint meghatároztuk a S_i részhalmazhoz tartozó szabályokat, akkor a szabályok segítségével meg kell határozni, hogy az összes többi részhalmazba tartozó elem melyik osztályba tartozna. Az eredmények követésére K-1 darab $ErrorResult_K$ nevű vektorban tartjuk nyilván, hogy helytelenül lett-e osztályozva az adott elem. A szabályok alkalmazása során három lehetséges kimenetel van: (1) Abban az esetben, ha a szabály

illeszkedik az elemre és helyesen megtudja határozni az osztályát 0 értéken hagyjuk az $ErrorResult_k$ értékét. (2) Ha a szabály illeszkedik az elemre, de helytelenül definiálja annak osztályát akkor pedig 1-re állítjuk. (3) Amennyiben egy szabály sem illeszkedett az adott sorra, akkor sem módosítjuk a változót, az továbbra is 0 marad. A megfelelő szabályok kiválasztásával ebben az esetben a szélsőséges, ám helyes értékek osztályozatlanok maradnak, mivel egy szabály sem illeszkedik rájuk, míg a zajos egyedekre pedig illeszkedni fognak a szabályok azonban helytelenül határozzák meg annak osztályát. Az egység végére érve tehát az adathalmaz minden egyes sorára megpróbáltuk K-1 alkalommal meghatározni a megfelelő osztályt a szabályok alkalmazásával. Az eredményeinket pedig a K-1 darab $ErrorResult_k$ nevű vektorokban tároljuk, melyeket egy ErrorResult nevű attribútumba összegzünk.

Index	$Attr_1$	$Attr_2$	$Attr_3$..	..$Attr_n$	Target label	Noisy label	$ErrC_1$	$ErrC_2$...	....$ErrC_k$	Error Count
1	37.027	52	5.765	123	1	1	0	0	0	0
2	65.964	62	5.346	435	1	0	1	0	1	7
3	35.346	26	6.567	324	0	1	1	1	1	9
4	35.345	45	0.4567	233	0	0	0	0	0	0
5	35.387	9	4.456	876	1	1	0	0	0	1

3.táblázat Az adathalmaz a szabályok alkalmazását követően

A 3. táblázatban az adathalmaz szabályok alkalmazását követő állapotát mutatja be, a 2 és 3 indexel rendelkező elemek zajosnak tekinthetők, és az algoritmus elfogja távolítani azokat a következő lépésben, ennek oka a K darab szabály alkalmazások során sokszor hibásan állapították meg a szabályok az adott elem címkéjét, mely hibaszám magasabb volt, mint amekkorát az E_{LTH} megenged.

3.3 Szűrések

A rendelkezésre álló adathalmaz ErrorResult attribútumát alapul véve határozhatjuk meg a modell által egész biztosan helyesnek, illetve zajosnak ítélt elemeket is. Abban az esetben, ha az ErrorResult attribútum értéke 0 értéken maradt az azt jelenti, hogy az eltérő szabályok halmazai rendre helyesen tudták meghatározni az osztályát. Azon elemeket tehát, melyek helyességében biztosak vagyunk kigyűjtjük és elhelyezzük a Tiszta adathalmazban (C). Egy előre meghatározott érték segítségével meghatározzuk

azt a E_{LTH} küszöbértéket, melyet meghaladó ErrorResult attribútum értékkel rendelkező elemeket Zajosnak definiálunk, és eltávolítjuk őket az adathalmazból (DS). A Zajos adathalmazba (F) kerülnek a zajosnak detektált elemek, ezen elemek tárolására csupán az algoritmus végén lévő kiértékeléskor lesz szükség, hogy mérni tudjuk a modellünk szűrési teljesítményét. Mindazonáltal, ha az adathalmazból eltávolítanánk az összes helyes elemet, így nem maradnának benne referencia elemek, melyek segítségével a további iterációkban megalkothatnánk a jó szabályok halmazát, valamint a kiegyensúlyozatlanság is, mint probléma megjelenne. A probléma megoldásaként a C adathalmaznak PB százaléknyi elemét visszahelyezzük az eredeti adathalmazba.

3.4 Kiértékelés:

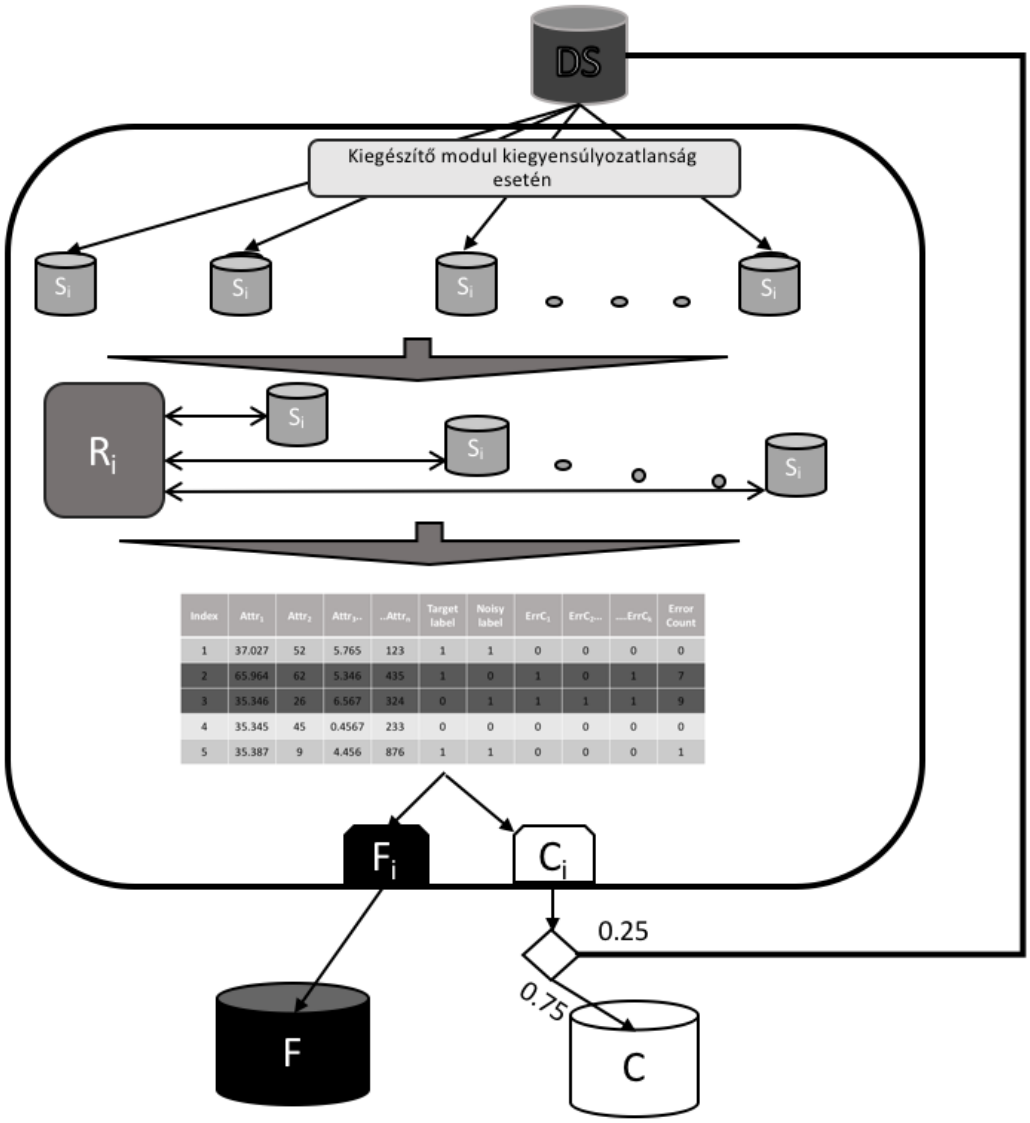
Az iteráció végeztével tehát három adathalmaz áll elő: a részben zajos, a tisztított, valamint a csupa zajos elemeket tartalmazó adathalmaz. A cél az lenne, hogy az iterációk egészen addig folytatódjanak, amíg az összes zaj kiszűrésre nem kerül, azonban könnyen korruptálódhat a modell és így végtelen ciklusba kerülne, ennek kiküszöbölése céljából egy külön kilépési feltételt határoztam meg. Legyen T_1 az aktuális iterációs során kiszűrt zajos elemek száma, továbbá T_0 az eggyel korábbi iterációban megtalált zajok száma. Amennyiben $T_1 < T_0 * 0{,}01$ abban az esetben modell működése leáll. A modell működését számos metrikával lehet mérni, a két legfontosabb azonban a detektált zajok aránya, valamint a zajszűrő modell hatása a komplett gépi tanuló rendszer működésére.

3.5 Kiegészítés: Kiegyensúlyozatlan adathalmaz kezelés

Zajszűrő modellek esetén kiváltképp fontos a kiegyensúlyozatlan adatok kezelése. A fejezetben bemutatott EPF modellem működése során is kiemelt figyelmet kapott a feladat. A zajszűrő modellbe készítettem egy kiegészítő részegységet, melyet kiegyensúlyozatlan adathalmazok esetén alkalmazok.

Kiegyensúlyozatlan adathalmazok kezelésére a 2.5 fejezetben bemutattam a legjelentősebb technikákat. A zajszűrési feladat megoldása során a felül mintavételezés tűnik a legjobb megoldásnak, alul mintavételezés jelentős információ veszteséggel járna, melyet a zajszűrési szakaszban nem engedhetünk meg. A túlmintavételezés során a kisebbségben lévő osztály elemeit a SMOTE módszer segítségével megnöveljük. A folyamat során fontos, hogy nem szabad a kisebbségben lévő adathalmaz elemeit jelentősen megnövelni, melynek oka, hogy a SMOTE alkalmazása során nem tudhatjuk,

hogy a minta elem valójában zajos volt-e vagy sem. Amennyiben rendelkezésre állnak historikus adatok korábbról, úgy meghatározható az ideálisra növelhető kisebbségben lévő adathalmaz aránya.



Index	Attr1	Attr2	Attr3..	...Attrn	Target label	Noisy label	ErrC1	ErrC2...	....ErrCk	Error Count
1	37.027	52	5.765	123	1	1	0	0	0	0
2	65.964	62	5.346	435	1	0	1	0	1	7
3	35.346	26	6.567	324	0	1	1	1	1	9
4	35.345	45	0.4567	233	0	0	0	0	0	0
5	35.387	9	4.456	876	1	1	0	0	0	1

11. ábra Az elkészített EPF működési folyamata

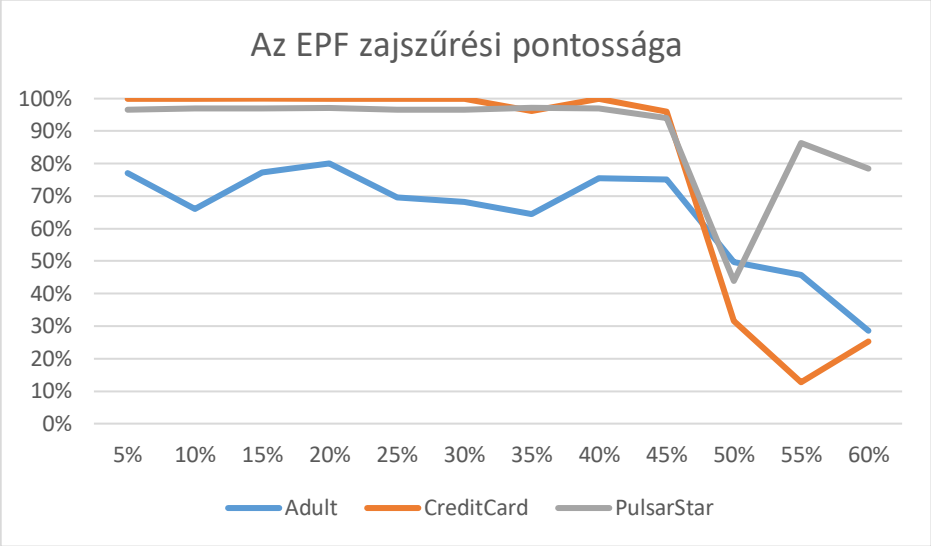
A 11. ábrán az EPF működése látható, az ábra röviden összefoglalja a szűrő működésének a főbb lépéseit és összefüggő képet ad róla.

3.6 Zajszűrési eredmények

Az EPF zajszűrő modell működését teszteltem három adathalmazon. Ahogy az 11 ábrán látható a modell rendkívül jó teljesítményt nyújt egészen addig, míg a helyes minták száma meghaladja a zajos elemekét. Ellenkező esetben az algoritmus a zajos elemeket

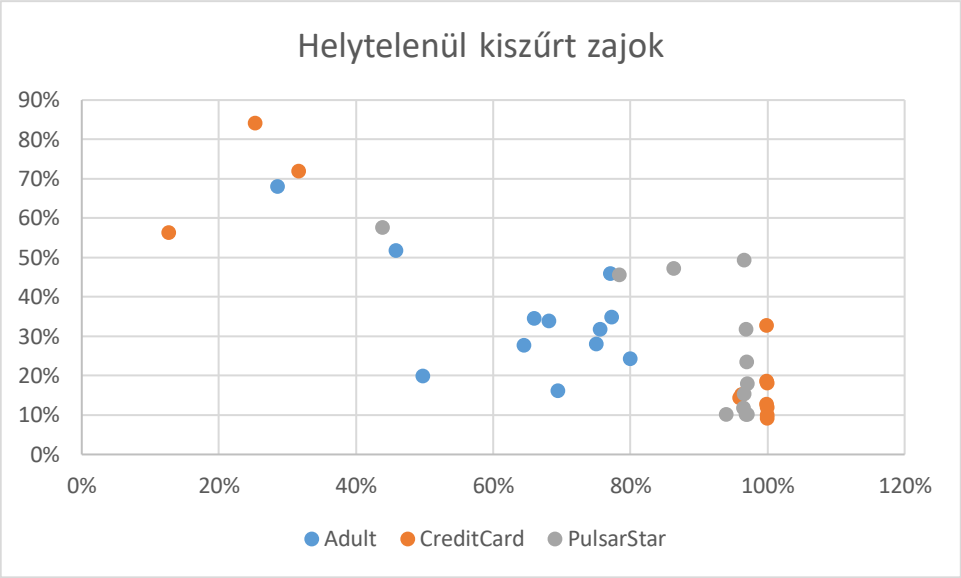
tekinti helyesnek és a külön-külön megalkotott szabályok már a zajos elemekre fognak illeszkedni.

A diagramon látható a modell teljesítménye, az X tengelyen található az adathalmaz zaj szintje, míg az Y tengelyen a megtalált zajos elemek aránya látható a 3 adathalmazon. Ezen zajszűrő modell teljesítménye jelentősen felül múlja a korábban bemutatott All kNN modellt 50 % -nál kisebb zajszintig.



12. ábra Az EPF zajszűrési pontossága

Zajszűrés esetén nem lehetünk teljesen biztosak abban, hogy a modellünk csak a valóban zajos egyedeket szűri ki. Bizonyos esetekben az olyan szélsőséges esetek is eliminálódnak, melyeket a modell nem tud megkülönböztetni megfelelőképpen a zajos egyedektől. A helytelenül kiszűrt esetek szám nagyban függ az adathalmaz karakterisztikájától is, minél kevesebb a szélsőséges elem, annál kevésbé fenyegetett az eliminálástól.



13. ábra A modell által helytelenül kiszűrt zajok aránya

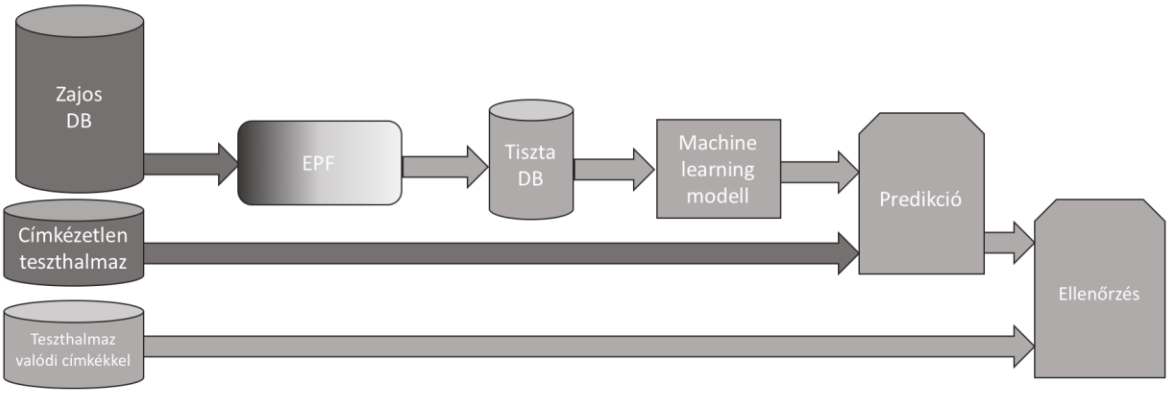
A 13. ábrán az X tengely mutatja az EPF modell által zajosnak talált esetek arányát a teljes adathalmazhoz viszonyítva, míg az Y tengely a zajként detektált elemek közt szereplő hamis pozitív elemek arányát.

4 Zajszűrési mérések különböző adathalmazokon

A 3. fejezetben bemutattam az általam elkészített és implementált zajszűrő modell felépítéseit és működésük hátterét. Azonban fontos megmutatni, hogy az elméletben helyesnek tűnő módszerek a különféle adathalmazokon való alkalmazás során is megfelelően teljesít és kiszűri az adathalmazban található zajok jelentős részét.

4.1 Komplex gépi tanuló rendszer

Jelen alfejezetben a zajszűrő modellek teljesítményének visszamérését fogom ismertetni, valamint az ehhez szükséges előkészületeket. Mint a fejezet bevezetőjében is említettem nehéz hozzájutni olyan zajos adathalmazhoz, melyen később visszamérhető a modellünk, ezért saját magam hoztam létre zajos adathalmazokat. Az 4.2 fejezetben bemutatott adathalmazokból készítettem különböző mértékben zajos példányokat. A zajmentes adathalmaztól egészen a 50 %-ban zajos adathalmazokig 5 %-os léptékkel hoztam létre új zajos adathalmazokat, melyeken alkalmaztam a modellem. Az adathalmaz eredeti osztálycímkéit azonban megtartottam, melyek az ellenőrzéshez elengedhetetlenek. A zajszűrő rendszerek a komplex gépi tanuló rendszer egy részegységét alkotják csak csupán. A mérések során adathalmazonként meghatároztam egy optimális gépi tanuló modellt a megfelelő paraméterekkel az osztályozás végrehajtására. Osztályozás lévén a RandomForest [6] modellt használtam predikcióra, azonban a modell hyperparaméterei a három adathalmaz esetén eltérőek voltak. A zajszűrő rendszerem kiértékelése a következő folyamatok során történt meg



14. ábra A komplex gépi tanuló rendszer működése

A komplex gépi tanuló rendszer működését szemlélteti a 14. ábra, mely során rendelkezésünkre áll egy zajos adathalmaz, és két teszthalmaz, egy címkézetlen, melyet a predikciókhoz használunk fel, míg a címkézett teszthalmazt az ellenőrzéshez. A zajos adathalmazon alkalmazva az EPF zajszűrő modellt előáll a tisztított adathalmaz a gépi tanuló algoritmus számára. A modell tanító mintaként felhasználja a tiszta adathalmazt, majd predikciókat készítünk a címkézetlen teszthalmazon. A rendszer kiértékelését az ellenőrzés során végezzük el a valós címkéket tartalmazó teszthalmazt felhasználva összevetjük a predikált illetve valós eredményeket.

4.2 Zajszűréshez felhasznált adathalmazok

Predicting Pulsar Star:

Az adathalmaz a High Time Resolution Universe Survey keretei közt lett létrehozva. A cél, hogy a 8 attribútum segítségével el tudjuk dönteni, hogy egy csillag pulzár csillagnak tekinthető-e. A pulzár csillagok olyan neutron csillagok melyek erős mágneses tevékenységei a földről is mérhetők. Az adathalmaz 17898 darab mintát tartalmaz, melyből 10% reprezentálja a pulzár csillagokat [17].

Adult Income Dataset:

Számos különböző amerikai társadalmi rétegből származó nagykorú személy adatait tartalmazza az adathalmaz. Az adathalmaz segítségével meghatározhatjuk azon személyes tulajdonságokat, attribútumokat, melyek jelentősen befolyásolják a fizetést. A

korábbi adathalmazokhoz hasonlóan jelen esetben is bináris osztályozás (éves szinten 50,000$-nál magasabb fizetés, vagy alacsonyabb) végezhető. Az adathalmaz 14 attribútumot tartalmaz, melyek demográfiailag képesek leírni egy személyt. Az adathalmaz 45225 darab mintát tartalmaz, melyben az osztályok aránya kiegyenlítettnek mondható [18].

Credit Card Fraud Detection:

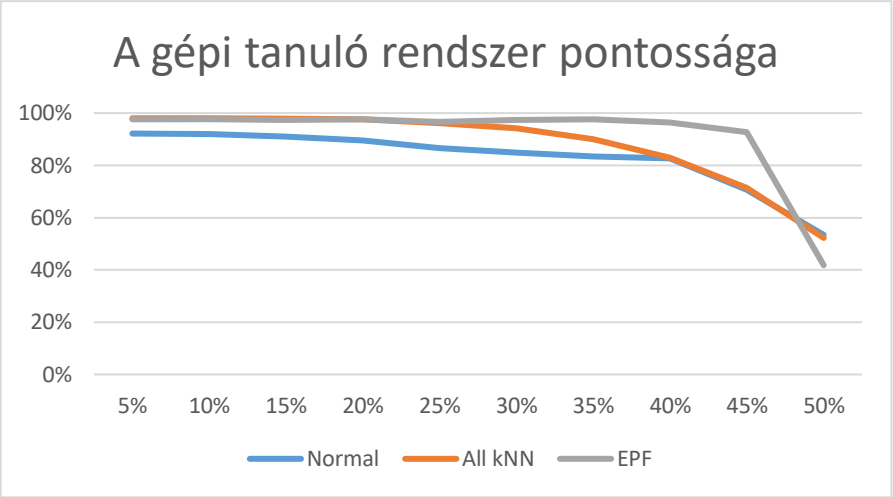
Az adathalmazt egy belgiumi Machine Learning kutató csoport szolgáltatta, a Credit Card Fraud Detection adathalmaz valós adatokat tartalmaz, melynek egyik jelentős hátránya, hogy a bank, valamint a partnerek személyes adatainak védelmében az adatokon PCA transzformációt végeztek. A transzformáció eredményeként a változók neveit V + sorszámra cserélték, értékeik pedig egy titkosított transzformációs függvény eredményei. Az adathalmaz 31 változót tartalmaz, melyből 30 numerikus és 1 kategorikus változó. Az adathalmaz 284807 darab tranzakciót tartalmaz, melyből 492-t soroltak a csalás kategóriába. A csalások száma 0,17% a teljes tranzakció számhoz viszonyítva. Az adathalmaz kiegyensúlyozatlanságát a modell építése, valamint a zajszűrés során kezelni kell a korábban ismertetett módszerek segítségével [16].

4.3 Mérési eredmények vizualizálása adathalmazonként

A 4.2 alfejezetben ismertetett adathalmazokon méréseket végeztem a zajszűrő modellek teljesítményének helyességével kapcsolatosan az egész rendszerre nézve. A mérés során a 4.1 alfejezetben bemutatott módszereket követtem végig, és a mérési eredményeket adathalmazonként publikálom.

Predicting Pulsar Star adathalmaz:

Az Pulsar Star adathalmazon végzett mérések következtében a gépi tanuló rendszer teljesítménye az alábbi ábrán megtalálható:



15. ábra A Pulsar Star adathalmaz pontossága

Az EPF teljesített a legjobban egészen egy bizonyos töréspontig, melytől kezdve teljesítménye jelentősen lecsökken. Az All kNN esetén is megfigyelhető a teljesítmény csökkenés, azonban annak mértéke egyenletes szemben a partitioning filterével. Mindkét modell alkalmazásával teljesítmény javulást értünk el az adathalmazon alkalmazva. A Normal modell esetén nem végeztem zajszűrést és így a modellem a zajos adatokon tanult, természetesen ennek a modellnek a pontossága a legalacsonyabb.

Az alábbi táblázat az komplex gépi tanuló rendszer eredményeit tartalmazza az adathalmaz zajszintje, valamint kiegyensúlyozatlansága függvényében.

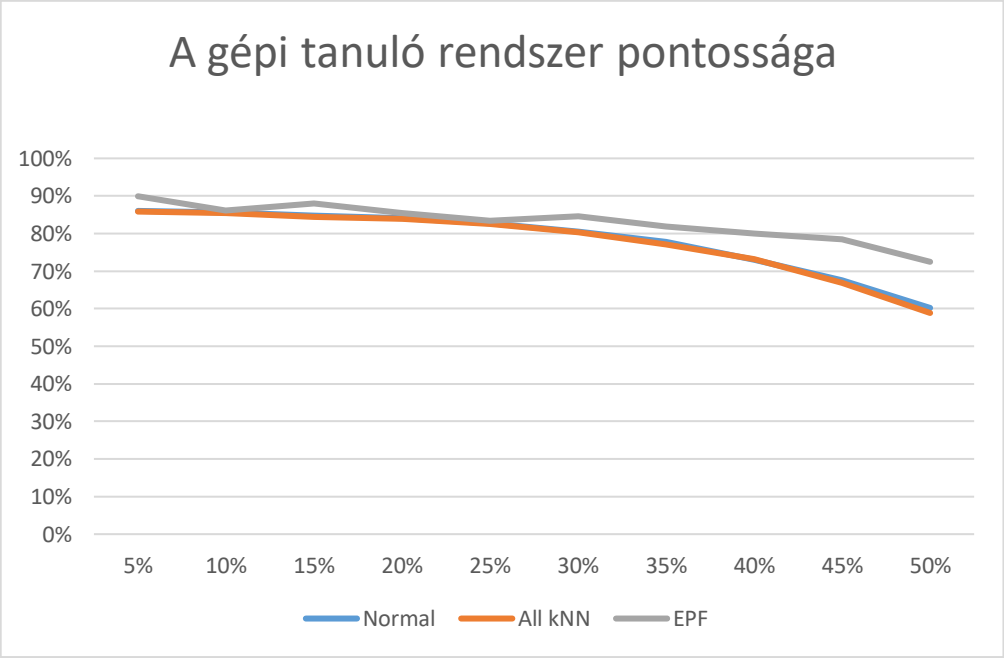
Zaj mértéke	C1/C2	Pontosság
10	10	0,963784109
	20	0,962942272
	50	0,975791434
20	10	0,956610801
	20	0,9716946
	50	0,97877095
30	10	0,951024209
	20	0,966294227
	50	0,97877095
40	10	0,895903166
	20	0,905629423
	50	0,966908752
50	10	0,433147114
	20	0,480446927
	50	0,508752328

4. táblázat: Pontosság a zaj és az osztályok viszonya alapján

A 4.táblázat első oszlopa szemlélteti a zaj mértékét a Pulsar Star adathalmazon, míg a második oszlop a két osztály arányát egymáshoz képest a harmadik oszlop pedig a komplex rendszer eredményét EPF zajszűrő segítségével. Az eredmények alapján a modell teljesítményére pozitív hatással van az EPF modell zajszűrése kiegyensúlyozatlan adathalmazokon is.

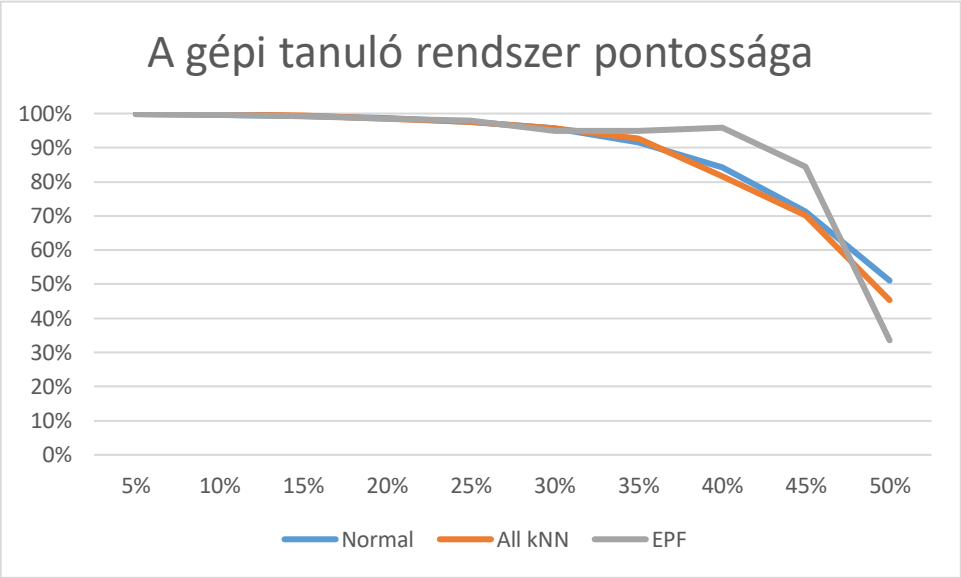
Adult Income adathalmaz:

Az Adult adathalmazon végzett mérések következtében a gépi tanuló rendszer teljesítménye az alábbi ábrán megtalálható, a modell jelen adathalmazon közel sem volt képes olyan magas teljesítmény elérésére osztályozás terén, mint ami a Pulsar Start és a CreditCard adathalmaz esetén megfigyelhető volt. Azonban a zaj szűrő modellek működésének hatását az alábbi diagram is jól tükrözi.



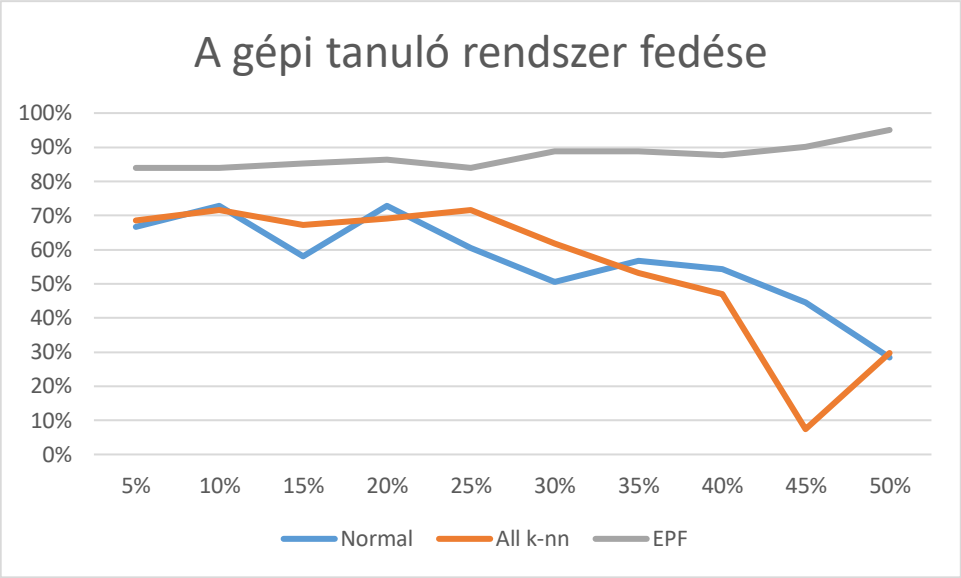
16, ábra Az Adult adathalmaz pontossága

Credit Card Fraud Detection adathalmaz:



17. ábra A Credit Card adathalmaz pontossága

A CreditCard adathalmazon alkalmazott modell pontossága a különböző zajszűrő módszerek felhasználásával. Az X tengelyen a tanuló adathalmaz zajszintje található, míg az Y tengelyen a modell végső pontossága. Osztályozási feladatok esetén a modellek hatékony teljesítményét megfelelően mutató indikátor a fedés, mely jelen esetben a csalásokra vonatkozik, valamint a ROC görbe alatti terület nagysága (azaz AUC). A következő diagramok ezen értékeket szemléltetik.



18. ábra Fedés a CreditCard adathalmazon

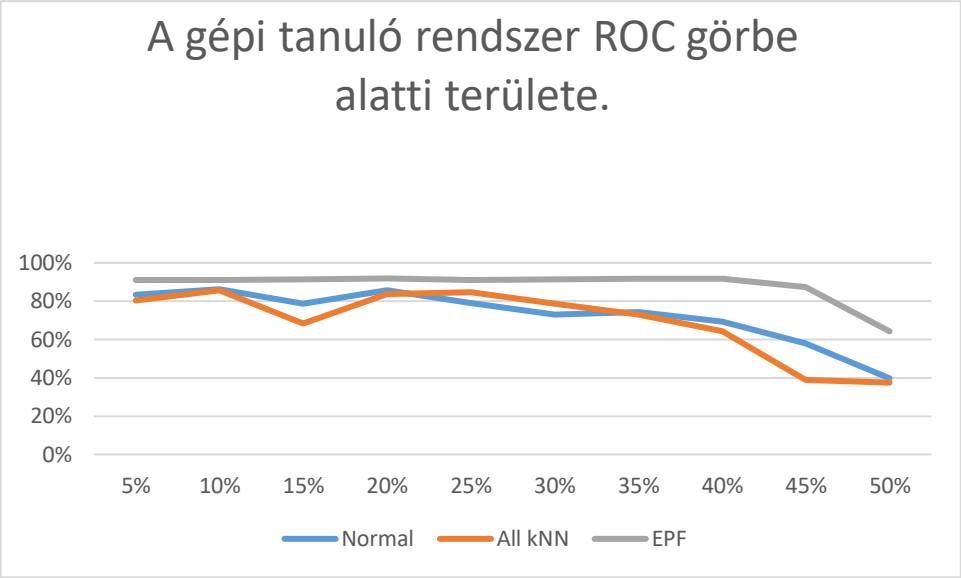
Az ábrán látható a modell fedése eltérő modellek alkalmazása esetén. A legjobb teljesítményt az EPF modell nyújtja, azonban rendkívül magas zaj szint esetén egyre több elemet szűr ki tévesen, és előfordulhat, hogy a tévesen eliminált elemek egy azon osztályból származnak. A modell fedése az alábbi képlet segítségével határozható meg, ahol TP a helyesen besorolt pozitívak, az FN pedig a helytelenül negatívnak osztályzott pozitívak számát jelenti.

$$Fedés = \frac{TP}{TP + FN}$$

A ROC görbe alkalmazásával grafikusan szemléltethető egy osztályozó teljesítménye. A ROC görbét ábrázoló diagram x tengelyén a hamis pozitív arány található, míg az y tengelyen az valós pozitív arányt. A görbén található pontok egy osztályozó modell eredményeinek felelnek meg. A görbe alatti terület nagysága (azaz AUC) számszerűen is megmutatja az osztályozó modell teljesítményét, melyet a következő képpen lehet meghatározni:

- X= Hamis pozitív arány $\left(\frac{FP}{(TN+FP)}\right)$

- Y= Valós pozitív arány $\left(\frac{TP}{(TP+FN)}\right)$



19. ábra AUC a Credit Card adathalmazon

A Pulsar Star adathalmazhoz hasonlóan a CreditCard adathalmaz is kiegyensúlyozatlannak tekinthető. Méréseket végeztem annak érdekében, hogy megvizsgáljam a zaj mérték és kiegyensúlyozatlanság mértékének a hatását a komplex rendszer pontosságára.

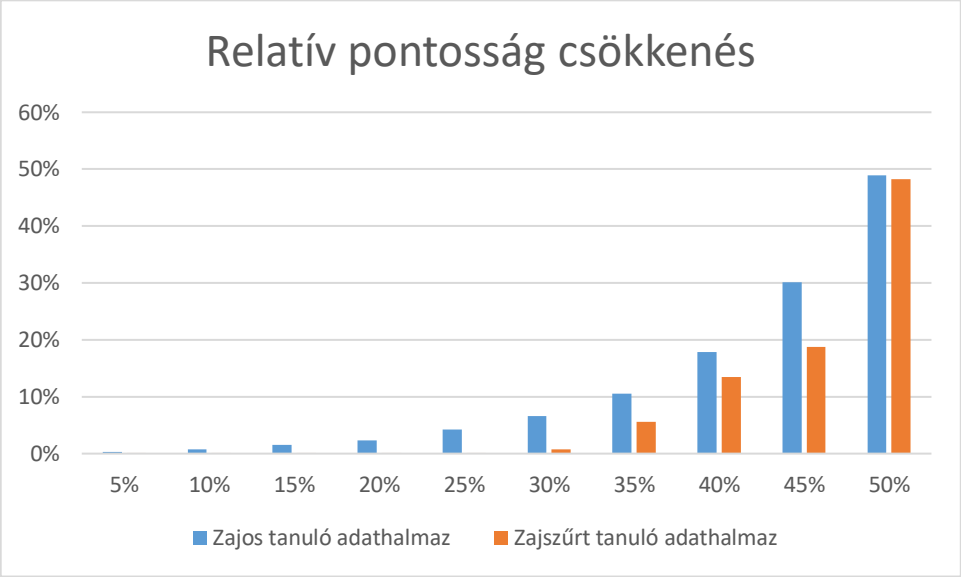
Zaj mértéke	C1/C2	Pontosság
10	10	0,963784109
	20	0,984450241
	50	0,989187199
20	10	0,981071429
	20	0,98344703
	50	0,989165329
30	10	0,981766653
	20	0,985929976
	50	0,988011637
40	10	0,946579053
	20	0,960724318
	50	0,983045746
50	10	0,357042536
	20	0,404745185
	50	0,453726926

5. táblázat: Pontosság a zaj és az osztályok viszonya alapján

Az 5. táblázat első oszlopa szemlélteti a zaj mértékét a Credit Card adathalmazon, míg a második oszlop a két osztály arányát egymáshoz képest a harmadik oszlop pedig a komplex rendszer eredményét EPF zajszűrő segítségével.

4.4 Mérési eredmények összegzése

Az adathalmazokon végzett mérések eredményei egyértelműen bizonyítják az EPF alkalmazásának előnyeit zajos adathalmazok esetén, ezt erősíti meg az alábbi diagram, mely a EPF-el szűrt tanuló adathalmaz és a zajos adathalmazzal tanított rendszer pontosságának relatív csökkenését mutatja.



20. ábra Relatív pontosság csökkenése zajos adathalmazok esetén

A 20. ábra X tengelye a tanuló adathalmaz zajosságát azonosítja, míg az Y tengely a zajos tanuló adathalmazzal tanított modell relatív pontosság csökkenését (RLA) szemlélteti. Megfigyelhető, hogy egészen 35%-ig az EPF modellt használva szinte eltekinthetünk az adathalmaz zajos eredetétől, mivel a modell kiváló működésének következtében a zajok pontosságra gyakorolt hatása elenyésző.

5 Összefoglalás

A kutatási munkám során megismerkedtem az osztályozási feladatok során jelentkező zajokkal, valamint azok csoportosításával. A zajok kezelését és a meglévő zajszűrő algoritmusok tanulmányozását követően a munkámat az All kNN és a partitioning filter részletes elemzésével folytattam. Saját zajszűrő modellem megalkotása során a partitioning filtert vettem alapul, és elkészítettem az extended partitioning filter zajszűrő modellt. Az általam elkészített modellben módosításokat és kiegészítéseket alkalmaztam az eredeti partitioning filter működéséhez képest, ezzel jó teljesítményt elérve kiegyensúlyozatlan adathalmazokon is. A modell helyességét mérésekkel támasztottam alá három különböző adathalmaz alkalmazásával, ahol két másik megoldással vetettem össze a modellemet. Mérések során bebizonyosodott, hogy az extended partitioning filter teljesített a legjobban.

A szakirodalomban számos cikk foglalkozik az adathalmazokban található zajok kezelésével, ám ezek nagy része az attribútumokat érintő zajokra irányul. A kutatási munkám azonban a kevésbé ismert címkézési hibákból eredő zajokkal és azok megfelelő kezelésével foglalkozott.

6 Irodalomjegyzék

[1] X. Zhu, X. Wu, *Class Noise vs. Attribute Noise: A Quantitative Study,* Artificial Intelligence Review 22 (2004) 177-210 doi: 10.1007/s10462-004-0751-8)

[2] X. Wu, X. Zhu, *Mining with noise knowledge: Error-aware data mining,* IEEE Transactions on Systems, Man, and Cybernetics 38 (2008) 917-932 doi: 10.1109/TSMCA.2008.923034):

[3] M.A. Hernández, S.J. Stolfo, *Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem,* Data Mining and Knowledge Discovery 2 (1998) 9-37 doi: 10.1023/A:1009761603038)

[4] J.R. QUINLAN C4.5 Programs for machine learning : https://link.springer.com/content/pdf/10.1007%2FBF00993309.pdf (1994)

[5] J.A. Sáez, J. Luengo, F. Herrera, Fuzzy rule based classification systems versus crisp robust learners trained in presence of class noise's effects: a case of study, in: 11th International Conference on Intelligent Systems Design and Applications (2011) 1229-1234 doi: 10.1109/ISDA.2011.6121827

[6] Bodon Ferenc: Adatbányászati algoritmusok : http://www.cs.bme.hu/~bodon/magyar/adatbanyaszat/tanulmany/adatbanyaszat.pdf (2010)

[7] Chan, P. K.-W. (1996). An extensive meta-learning approach for scalable and accurate inductive learning, Ph.D Thesis, Columbia University.

[8] I. Tomek. An Experiment With The Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man and Cybernetics 6:6 (1976) 448-452 doi: 10.1109/TSMC.1976.4309523]

[9] Eliminating Class noise in large datasets :https://pdfs.semanticscholar.org/2b91/3ec77958e39f7eda3812a5d08e10e764f095.pdf

[10] .Jose.A. Sáez, J. Luengo, F. Herrera, Evaluating the classifier behavior with noisy data considering performance and robustness: The Equalized Loss of Accuracy measure. Neurocomputing, in press (2015), doi: 10.1016/j.neucom.2014.11.086].

[11] Introduction to noise in data mining: http://sci2s.ugr.es/noisydata

[12] C.E. Brodley, M.A. Friedl. Identifying Mislabeled Training Data. Journal of Artificial Intelligence Research 11 (1999) 131-167 doi: 10.1613/jair.606]

[13] José A. Sáez, M. Galar, J. Luengo, F. Herrera, INFFC: An iterative class noise filter based on the fusion of classifiers with noise sensitivity control. Information Fusion, 27 (2016) 19-32, doi: 10.1016/j.inffus.2015.04.002.

[14] José A. Sáez, J. Luengo, J. Stefanowski, F. Herrera, *SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering.* Information Sciences, 291 (2015) 184-203, doi: 10.1016/j.ins.2014.08.051.

[15] Imbalanced learn: *http://contrib.scikit-learn.org/imbalanced-learn/stable/index.html*(2018.03.27)

[16] Credit Card fraud detection: https://www.kaggle.com/mlg-ulb/creditcardfraud

[17] Predicting a pulsar star: https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star/home

[18] Adult Income Dataset :https://www.kaggle.com/wenruliu/adult-income-dataset