Önálló laboratórium beszámoló

Távközlési és Mesterséges Intelligencia Tanszék

Készítette:	**Czotter Benedek**
Neptun-kód:	**TFB4FY**
Ágazat:	**Információs rendszerek specializáció, TMIT ágazat**
E-mail cím:	**czottibeni@gmail.com**
Konzulens(ek):	**Dr. Szűcs Gábor, Németh Marcell**
E-mail címe(ik):	**szucs@tmit.bme.hu, nemethm@tmit.bme.hu**

Ensemble szegmentáló módszerek idősoros adatokon

Feladat

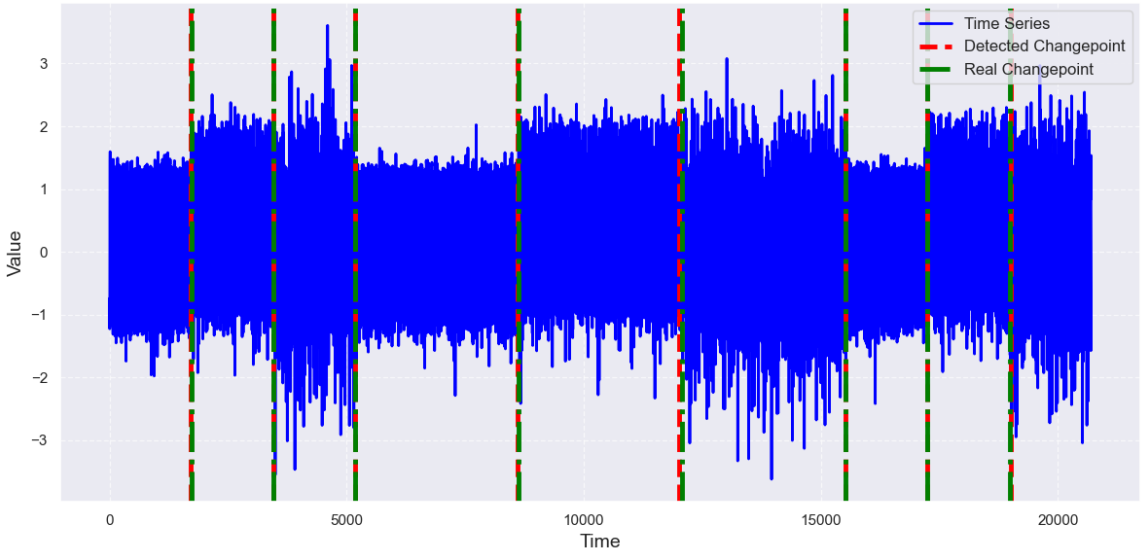
Az idősorok szegmentálása kulcsfontosságú probléma adatelemzési feladatokban. Idősorok szegmentálása alatt az idősor releváns szakaszokra való felbontását értjük, annak érdekében, hogy azonosítsuk a mintázatokat, változási pontokat vagy eltérő viselkedésű szakaszokat az adatsoron belül. Ehhez kapcsolódóan a feladat az volt, hogy ha van több szegmentáló módszer (vannak köztük jobbak és kevésbé jók), akkor hogyan lehetne őket kombinálni, hogy egy jobb (pontosabb) szegmentáló módszert kapjunk? Azaz van k db kész módszer a szakirodalomban, akkor egy ensemble módszer hogyan tud majd dönteni a szegmentálási feladatban?

2024/2025. 2. félév

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1 Bevezető

Az idősorok szegmentálása kulcsfontosságú probléma adatelemzési feladatokban. Idősorok szegmentálása alatt az idősor releváns szakaszokra való felbontását értjük, annak érdekében, hogy azonosítsuk a mintázatokat, változási pontokat vagy eltérő viselkedésű szakaszokat az adatsoron belül [1]. Mindez segítheti az elemzést, előrejelzést vagy döntéshozatalt. Számos idősor-szegmentálási módszer létezik, amelyeket különböző területeken használnak, mint például pénzügyi előrejelzések, orvosi diagnosztika vagy szenzor adatok elemzése. Jelen munka célja olyan ensemble módszerek kialakítása, amely több meglévő idősor-szegmentálási algoritmus kombinálásával, az egyes algoritmusok erősségeinek és gyengeségeinek feltérképezésével pontosabb eredményt biztosít.



1. ábra - Egy idősor szegmentálása CLaSP algoritmussal a detektált (piros) és valós (zöld) változáspontok jelölésével.

1.2 Elméleti összefoglaló

Az idősorok szegmentálására különböző megközelítések léteznek, beleértve a statisztikai alapú, klaszterezési, valószínűségi modellekre épülő és neurális hálózatokon alapuló megoldásokat. Az ensemble tanulás lényege, hogy több független modell döntését kombináljuk, ezáltal csökkentve az egyes modellek hibáinak a hatását, így egy robosztusabb, jobb általánosító képességgel rendelkező modellt kapjunk. Jelen dolgozat két eltérő megközelítést mutat be idősorok szegmentálására alkalmas ensemble modellek létrehozására, a modellek és egyes algoritmusok teljesítményének vizsgálatából átfogó képet alkotva arról, hogy adott típusú idősorhoz, milyen szegmentáló algoritmust célszerű választani a minél nagyobb pontosság elérése érdekében. Célom felhívni a figyelmet arra, hogy az adatok előfeldolgozása, az adathalmaz statisztikai mutatóinak pontos ismerete kulcsfontosságú szerepet tölt be az adatelemzési problémák területén, mindez megkönnyíti a megfelelő algoritmus kiválasztását, így lehetővé teszi pontosabb predikció elérését.

1.3 A munka állapota, készültségi foka a félév elején

A félév elején Dr. Szűcs Gábor és Németh Marcell konzulenseimmel meghatároztuk az elvégzendő feladatot. Segítségükkel elkészítettem a féléves munkatervet, amely mindvégig segítette a féléves munka strukturált megvalósítását. A munkaterv alapján az első hetekben

a szakirodalmi háttér feldolgozására és az alkalmazni kívánt szegmentálási algoritmusok kiválasztására koncentráltam. Ehhez számos friss tanulmányt elolvastam az idősor szegmentáló algoritmusokról, és a számomra releváns megközelítések közül kiválasztottam néhányat további vizsgálatra, melyekről a következő fejezetekben bővebben írok.

2. Az elvégzett munka és eredmények ismertetése

2.1 State-of-the-art idősor szegmentáló módszerek

Az ensemble modellek bemutatása előtt fontosnak tartom, hogy bemutassam a state-of the-art idősor szegmentáló módszerek közül az általam is kipróbált algoritmusokat, hiszen mindet optimalizálnom kellett, illetve szeretnék egy átfogó képet adni a későbbi tervezői döntéseim okáról, hogy egyértelműen kiderüljön, miért is ezeket az algoritmusokat választottam. A jelenleg is alkalmazott idősor szegmentáló módszereket négy nagy csoportba lehet sorolni. Igyekeztem mindegyikből kipróbálni algoritmusokat, és a későbbi ensemble modelleimben is mindegyikből használni.

Az első csoport a statisztikai és szabályalapú módszerek, melyek az idősorban található töréspontok (változáspontok) detektálására épülnek. Gyakran dinamikus programozási, Bayes-féle vagy optimalizációs technikákat alkalmaznak. Általában gyors futásidejűek és jól interpretálhatóak. Ebből a kategóriából számos algoritmust kipróbáltam, és használtam fel a későbbiek során ensemble modellekben is.

Az első ilyen algoritmus a Pruned Exact Linear Time (PELT) [10] volt, mely egy hatékony módszer idősorok változáspont detektálására. Az algoritmus által készített szegmentálás pontossága nagyban függ a büntetési paraméter (penalty) helyes megválasztásától. A penalty azt adja meg, hogy mennyire érzékenyen reagál a PELT az idősorban bekövetkező változásokra. Ennek optimalizálására egy tartományon belül egyenletesen elosztott büntetési értékeket teszteltem, és minden esetben kiszámoltam a szegmenshez tartozó költséget, amit a szegmens hosszának négyzeteként definiáltam. A megfelelő értéket a KneeLocator [7] algoritmus segítségével határoztam meg, ami visszaadja az optimális könyökpontot a költségfüggvény görbéjén. A túlzott szegmentálás elkerülése érdekében a minimális szegmens hosszúságot az idősor hosszának egyhetedére állítottam, tehát maximum 7 szegmenset engedélyeztem.

A következő algoritmus a Binary Segmentation (BinSeg) [11] volt, ami fokozatosan osztja fel az idősort szegmensekre. A módszer egyik kritikus eleme a szegmentációs pontok számának megfelelő meghatározása. Ennek az értéknek az optimalizálását végeztem el oly módon, hogy vettem egy maximális szegmens számot (n) az idősor hosszával arányosan, majd 1-től n-ig kiszámoltam az algoritmus összesített költségfüggvényét, amely azt méri, hogy az adott szegmentáció mennyire illeszkedik az idősor szerkezetéhez. Az optimális szegmens számot hasonlóan a PELT-nél ismertetett módon a KneeLocator [7] függvénnyel határoztam meg a költségfüggvény alapján.

Következő algoritmus a Bottom-Up [9], amely kezdetben az idősor teljes tartományát egyetlen szegmensként kezeli, majd fokozatosan finomítja a felosztást. A módszer megpróbálja az adatok szerkezetének megfelelően kialakítani a felosztást. Hasonlóan a BinSeg algoritmushoz, itt is a maximális szegmens szám paramétert optimalizáltam. Ennek végrehajtása teljes egészében megegyezik a korábbiakban bemutatott módszerrel.

Ebből a kategóriából az utolsó választott algoritmus a Window-Based szegmentációs algoritmus [1], ami egy csúszóablakos megközelítést alkalmaz az idősor felbontására. Ennek a módszernek az erőssége, hogy olyan idősorokon is jól használható, ahol a változások lokálisan észlelhetők, de az egész struktúra globálisan nem feltétlenül ismert. Az ablakméretet optimalizálása közben arra jutottam, hogy a maximuma az ablakméretnek legyen, vagy az idősor hosszának a 20-ad része, vagy 5. A változáspontok maximális számát

(n) az idősor hossza és az ablakméret alapján határoztam meg oly módon, hogy az legyen a minimuma, vagy 15-nek, vagy az idősor hosszának két ablakméreted részének. Ezt követően 1-től n-ig minden szegmens számhoz meghatároztam a változáspontok által meghatározott szegmensek hosszának a varianciáját. Az optimális szegmens számot hasonlóan a korábbiakhoz a KneeLocator [7] végzi, azzal a különbséggel, hogy most a varianciákból kapott görbe könyök pontját keresi. Mind a négy bemutatott algoritmus esetén L2 kernelt használtam. Az L2 kernel működése azon alapszik, hogy a kapott szakaszokon belül a lehető legkisebb négyzetes eltérést érjük el az adott szakasz átlagától.

Az idősor szegmentáló módszerek második kategóriája a klaszterezési és tanulási alapú módszerek. Ebből a kategóriából futtattam a Dynamic Time Warping KMeans (DTW KMeans) algoritmust [1], ami ellentétben az euklideszi távolságra épülő hagyományos KMeans-től, a dinamikus idővetemítést (DTW) használja távolságmértékként. DTW KMeans algoritmust azonban csak kipróbálás szintjén használtam, mivel a DTW csak egyenlő hosszú szegmenseket tud előállítani. Ellenben a szintén ebbe a csoportba sorolható, az egyik, ha nem a legpontosabb szegmentáló módszerrel, a Classification Score Profile-val (CLaSP), amelynek a működése azon alapszik, hogy az idősor különböző lehetséges vágási pontjainál két részre osztja az idősort, majd megvizsgálja, mennyire jól különböztethetők meg ezek a szegmensek egy osztályozóval [2]. A kapott klasszifikációs pontosság egy profilként értelmezhető az idő függvényében, melynek lokális maximumai jelenthetnek változáspontot. Ennek a módszernek a használatához egy jól optimalizált keretrendszert használtam, amelyet a claspy Python könyvtárból importáltam [3]. Az innen elérhető modellen saját optimalizációt nem kellett végeznem. A hivatkozott cikkek ([2], [3]) részletesen ismertették a CLaSP működési elvét, melyeket alaposan tanulmányoztam munkám során.

A harmadik csoportja az idősor szegmentáló módszereknek a mélytanulás alapú algoritmusok. Ebben a kategóriában lehet említést tenni Long Short-term Memory-ról (LSTM), transzformerekről, konvolúciós neurális hálókról (CNN) és Temporal Convolutional Networks-ről (TCN) ([4], [12]). Választásom utóbbira esett, így készítettem és megpróbáltam optimalizálni egy TCN alapú autoencodert [4], amely az adatok rekonstrukciós hibáját felhasználva detektálja a jelentős változáspontokat. A modell célja, hogy az idősor egy tömörített reprezentációját tanulja meg az encoder-decoder architektúra segítségével. Az encoder-t több dilatált konvolúciós rétegből építettem fel, amelyek egyre tömörebb reprezentációt képeznek az idősorból. Reziduális kapcsolatokat is alkalmaztam a mély hálózatok tanulásának elősegítésére. A decoder-t az encoder struktúrájának tükörképeként készítettem el. A modell tanítása során Mean Squared Error (MSE) veszteségfüggvénnyel és Adam optimalizálóval optimalizáltam. A tanulás hatékonyságának fenntartása érdekében ReduceLROnPlateau tanulási ráta csökkentő ütemezőt használtam. A változáspontok detektálása a rekonstrukciós hiba értékének elemzésén alapszik. Egy gördülő ablakos módszert alkalmaztam, amely a hiba átlagát és szórását vizsgálja. Ha az aktuális hiba meghaladja az átlag és a szórás egy küszöbértékkel vett szorzatának összegét, akkor azt a pontot változási pontként azonosítottam.

Az idősor szegmentáló módszerek negyedik és egyben utolsó csoportja, ami egyben az önálló laborom témája is, az ensemble szegmentáló módszerek. Ezek a módszerek több algoritmust kombinálnak a pontosság és robusztusság növelése érdekében. Például egy ensemble modell kombinálhat statisztikai, klaszterezési és mélytanulási módszereket, így csökkentve az egyedi modellek gyengeségeit, és felerősítve a modell általánosító képességét. Rugalmas és hatékony módszernek tűnhet többféle modell kombinálása, azonban a megfelelő modellek kiválasztása, a modellek súlyozása, az eltérő predikciók összehangolása egy komplex és kihívást jelentő feladat. Azonban ensemble modellek futtatása nagy mértékben növeli a predikciók előállításához szükséges számítási kapacitást.

A következő fejezetben részletesen bemutatom az általam készített ensemble modelleket, az általuk elért eredményeket, és össze is hasonlítom őket egyéni state-of-the-art megoldásokkal.

Statisztikai és szabályalapú módszerek	Klaszterezési és tanulási alapú módszerek	Mélytanulás alapú módszerek
Pruned Exact Linear Time (PELT)	Dynamic Time Warping KMeans (DTW KMeans)	Temporal Convolutional Network (TCN) Autoencoder
Binary Segmentation (BinSeg)	Classification Score Profile (CLaSP)	Long Short-term Memory (LSTM)
Bottom-Up		Konvolúciós neurális háló (CNN)
Window-Based		

1. táblázat - Az idősor szegmentáló algoritmusok csoportosítása

Ahogy azt az 1. táblázat is még egyszer összefoglalja idősorok szegmentálására alkalmas state-of-the-art algoritmusoknak három kategóriáját különböztetjük meg. A negyedik kategória alatt pedig ezeknek az algoritmusoknak együtteseit értjük. Az egyes kategóriákba tartozó algoritmusok bonyolultságáról elmondható, hogy általánosságban a statisztikai és szabályalapú módszerek a legegyszerűbbek és a mélytanulási algoritmusok a legbonyolultabbak, amely az algoritmusok futásidejében és a szükséges számítási kapacitásban is megmutatkozik. Minél bonyolultabb egy algoritmus, általában annál hosszabb futásidővel kell számolni, ami egy fontos szempont akkor, amikor ensemble modellekhez választjuk ki a használt algoritmusokat.

2.2 Ensemble szegmentáló módszerek

2.2.1 Változáspont detektálás hierarchikus klaszterezéssel

Idősorok ensemble modellel való szegmentálására az első megközelítés, amit alkalmaztam, egy, már egyszerűbb osztályozási problémáknál is alkalmazott, ensemble technikára próbáltam visszavezetni, amely a majority voting, azaz többségi szavazás. Ennek a technikának a lényege, hogy a különböző modellek adott bemenetre előállított predikcióit összeveti, és a végső döntést az alapján hozza meg, hogy a legtöbb modell milyen kimenetet adott. Ez a módszer különösen akkor hatékony, ha a modellek eltérő szempontok alapján hozzák meg döntéseiket, hiszen így a zaj vagy egyedi hibák hatása csökkenthető. Ezt a módszert próbáltam átültetni nem felügyelt tanulási környezetbe idősorok változáspontjainak detektálásához.

Egy idősort hatékonyan lehet reprezentálni, mint egymást követő időpontok, és hozzájuk tartozó értékek együttese. Változáspontok reprezentálásához előbbit, az időpontokat választottam, így egy modell kimenete egymást követő időpontok lesznek. Abban az esetben, ha veszek n modellt, ez n darab k_1, k_2, ..., k_n hosszú időpont sorozatot jelent, ahol k_i egy tetszőleges pozitív egész szám, melyet ahogy korábban már említettem bizonyos modellekhez pontosan meg lehet adni, más esetekben dinamikusan határozódik meg. Fontos azt is megjegyezni, hogy eltérő modellek eltérő számú változáspontot is detektálhatnak.

Miután mind az n modell elvégezte a predikciót, az így kapott n darab időpont listát,

összefűztem, majd pedig ezen az összefűzött listán végeztem hierarchikus klaszterezést. Ehhez a scipy.cluster.hierarchy.linkage() függvényét használtam, melyhez ward és single módszert is kipróbáltam, de végül a single mellett döntöttem, mivel amíg a ward egy variancia minimalizáló stratégiát követ, addig a single a klaszterek legközelebbi pontjai közötti távolság alapján hozza meg az összevonási döntést, ami jobban illeszkedik az egydimenziós, időpontokból álló lista klaszterezéséhez. A klaszterek szétválasztásához szükséges távolság küszöböt oly módon határoztam meg, hogy több hierarchikus klaszterezési szintet is megvizsgáltam, és az inkonzisztencia statisztikák alapján [8] választottam ki a megfelelő küszöbértéket. Ennek kiválasztására azzal a heurisztikával éltem, hogy azon a szinten célszerű elvégezni a vágást, és ezzel megkapva a leginformatívabb szétválasztási pontokat, ahol a legnagyobb az inkonzisztencia-szórás. Végül a távolság küszöb értékét az adott szint inkonzisztencia koefficiens átlagának és szórásának összegeként határoztam meg. Ez az inkonzisztencia koefficiens mutatja meg mennyire szokatlan vagy nem homogén egy adott klaszter. A 2. ábra pszeudo kódján látható inconsistency[:,3] jelenti az inkonzisztencia koefficiens értékét a scipy.cluster.hierarchy.inconsistent metódusa által visszaadott n x 4-es mátrixban.

```
function find_optimal_threshold(Z, max_depth):
    max_variation ← 0
    best_threshold←None

    for depth from 1 to max_depth:
        inconsistency ← inconsistent(Z, depth)
        values ← inconsistency[:, 3]
        threshold ← mean(values) + std(values)
        variation←std(values)

        if variation > max_variation:
            max_variation ← variation
            best_threshold←threshold
    return best_threshold
```

2. ábra - A hiererchikus klaszterezés optimális vágás algoritmusának pszeudokódja.

Miután a változáspontokat klaszterekbe soroltam, minden klaszterhez választottam egy reprezentatív pontot, amit az adott klaszterbe tartozó változáspontok átlagaként határoztam meg. Ez biztosítja azt, hogy az egyes szegmentálási algoritmusok által feltárt közeli változáspontok, egyetlen változáspontként jelenjenek meg a végeredményben.

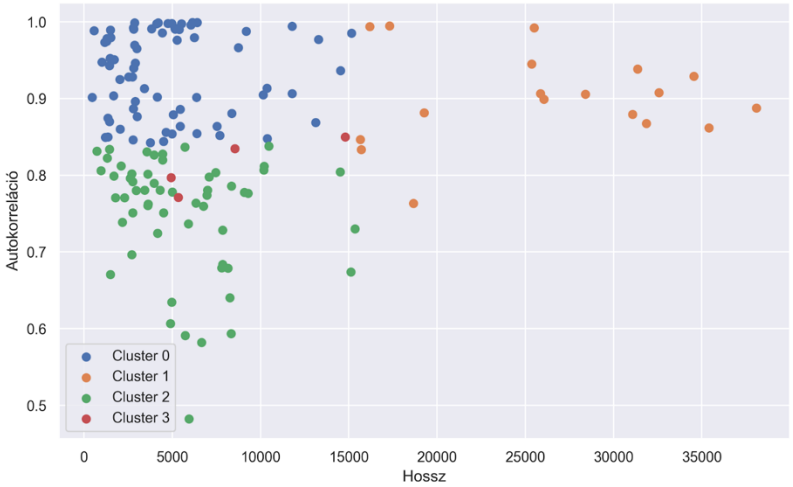
Számos algoritmus kombinációját kipróbáltam ezzel a megközelítéssel. Az algoritmusok kiválasztásánál kétféle szempontot tartottam szem előtt. Az első esetben megpróbáltam hasonló működési elven, vagy hasonló bonyolultságú modelleket kiválasztani. Így például készítettem egy modellt a PELT, BinSeg és BottomUp hármasból, de a Window, BinSeg, BottomUp triót is kipróbáltam. A másik megközelítésem az volt, hogy eltérő bonyolultságú és eltérő működési elven alapuló modell együtteseket hozzak létre. Ezalapján választottam a CLaSP, BinSeg és Window módszereket, hiszen a CLaSP egy jóval bonyolultabb, és más megközelítéssel rendelkező algoritmust valósít meg, mint a másik kettő.

2.2.2 Klaszterenként végzett predikció

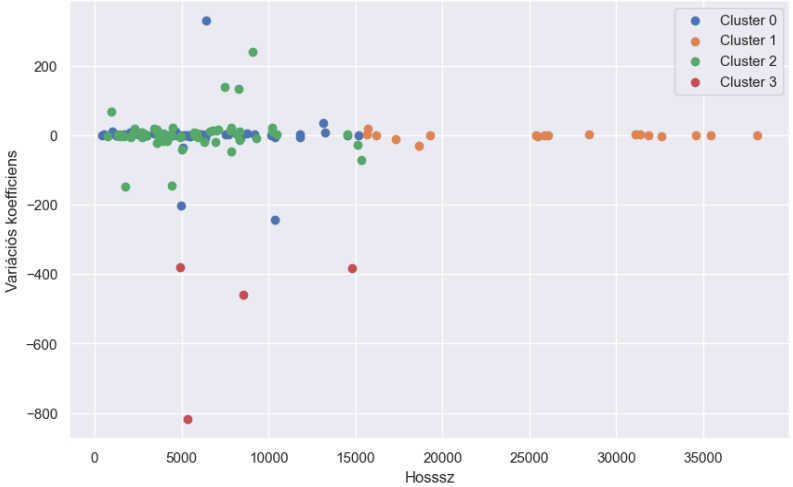
A második megközelítés, amit idősorok ensemble modellekkel való szegmentálására kidolgoztam, megpróbálja kihasználni az egyes algoritmusok erősségeit. Egy idősort nagyon sok statisztikai mutatóval lehet jellemezni, amelyek más és más oldalról próbálják

meg leírni az idősor viselkedését és tulajdonságait. Ezek a statisztikai jellemzők nagyban befolyásolják az egyes modellek teljesítményét, amikor szegmentálásra kerül a sor, hiszen, mint azt korábban az egyes algoritmusok optimalizálásának bemutatásakor említettem, az idősor hossza befolyásolhatja az ablakméretet, de az idősor varianciája, autokorrelációs értéke és egyéb mutatói, mind-mind szerepet játszanak az algoritmusok döntéshozatalában.

A módszerem célja megtalálni idősorok azon csoportjait az egyes algoritmusokhoz, amelyeken a legjobb teljesítményt nyújtják. Ehhez a rendelkezésre álló adathalmazokat, a Time Series Segmentation Benchmark-ot (TSSB), mely 75 darab idősort tartalmaz, [5] és Human Activity Segmentation Challenge-t (HAS), amely pedig 250 idősorból áll [6] felosztottam tanító és teszt halmazra 60-40%-os arányban, így a TSSB tanító halmaza 45, teszt 30, a HAS tanító halmaza 150, teszt pedig 100 darab idősorból állt, majd a tanító halmazban szereplő idősoroknak kinyertem különböző statisztikai jellemzőit, amelyek alapján KMeans klaszterezéssel próbáltam kialakítani hasonló karakterisztikát mutató idősorok csoportjait. A klaszterezés elkészítéséhez az egyes idősoroknak megvizsgáltam a hosszát, trendjét, autokorrelációs értékét, illetve a variációs koefficienst, ami megadja az idősor szórásának és átlagának hányadosát. Ez azért hasznos, hogy különböző skálájú idősorokat tudjunk összehasonlítani. Ezenkívül vizsgáltam az idősorok mozgó ablakos szórásának átlagát is, amihez az ablakméretet 10 egység hosszúnak állítottam be. Az optimális klaszterszám meghatározásához ezúttal is könyök pont keresést alkalmaztam.



3. ábra - Az elkülönülő idősor klaszterek a hossz és autokorreláció függvényében.



4. ábra - Azon néhány idősor elkülönülése, melyek variációs koefficiense kiugró.

Ahogy az a 3. és 4. ábrán is látható, az ideális klaszterszámot négynek választottam. Ezután megvizsgáltam a rendelkezésre álló modellek teljesítményét minden klaszteren, így megkaptam, hogy melyik klaszteren melyik idősor szegmentáló algoritmus teljesít a legjobban. A vizsgált algoritmusok között volt a CLaSP, BinSeg, BottomUp, Window, illetve a TCN alapú autoencoder. Tehát ezen öt algoritmusból kerestem a négy klaszterhez a legmegfelelőbb algoritmusokat.

Ezt követően a teszthalmazban szereplő idősorokból is kinyertem a vizsgált statisztikai jellemzőket, majd besoroltam őket ez alapján a létező klaszterek egyikébe. A szegmentálást pedig azzal az algoritmussal végeztem, amelyikről a tanító fázisban kiderült, hogy az adott klaszteren a legjobban teljesít, amit a következő fejezetben szintén részletezni fogok.

2.3 Modellek kiértékelése

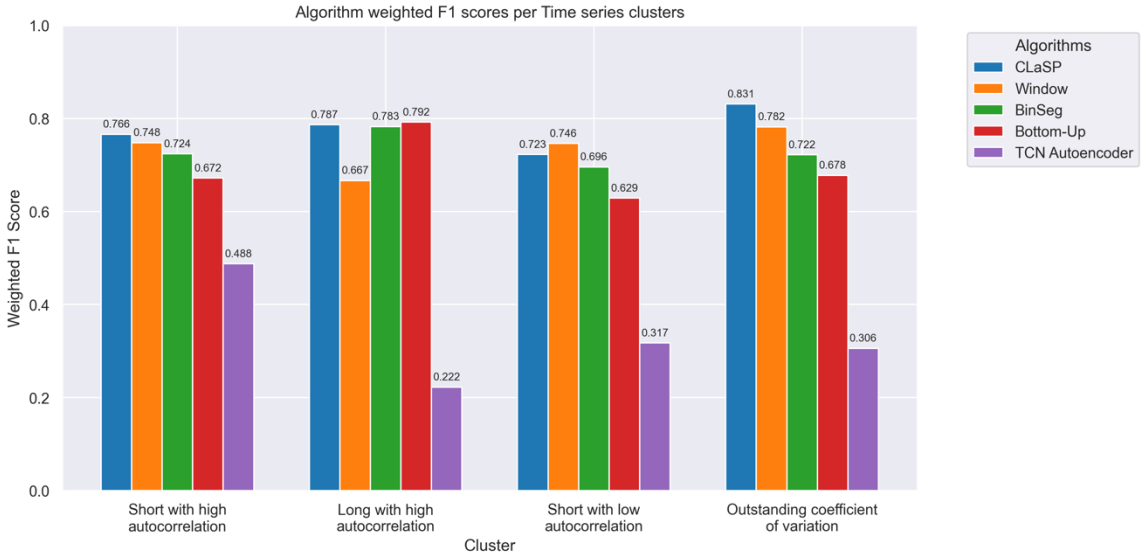
A különböző modellek kiértékelésére kétféle metrikát vizsgálok. Az első az F1-score, amit az idősorok hosszával még súlyozok, mivel minél hosszabb egy idősor, annál magasabb a hibázási lehetőség. Ez az F1 érték az alapján számítódik, hogy az egyes algoritmusok predikciója mennyire jól illeszkedik a valós változáspontok helyzetéhez egy megengedett margin hibával. A másik metrika, amit alkalmaztam egy úgynevezett covering értéket ad vissza. Ez azt jelent, hogy a modell által visszaadott szegmensek mennyire jól fedik le a tényleges szegmenseket. Mindkét kiértékelési mód a claspy könyvtárban található meg, azzal a különbséggel, hogy az F1 érték ebben a könyvtárban nincs alapértelmezetten súlyozva az idősorok hosszával, ezt én gondoltam hasznos kiegészítésnek. Ehhez az egyes idősorokon elért F1 értéket megszoroztam az idősor hosszával, majd így szummáztam őket, végezetül pedig leosztottam az összes idősor hosszának összegével.

A kiértékeléshez és tanításhoz a már korábban is említett két adathalmazt a TSSB-t [5] és HAS-t [6] használtam fel. Mindkettő változó hosszúságú és változó karakterisztikájú idősorokat tartalmaz. A TSSB 75 míg a HAS 250 előre szegmentált idősort tartalmaz, így könnyen visszamérhető a modellek predikcióinak pontossága. Azon modellek esetén, ahol tanító fázis is szerepelt az algoritmusban, a tanítást az adathalmazok 60%-án, kiértékelését pedig a maradék 40%-on végeztem. A konzisztencia megőrzése érdekében azokat az algoritmusokat is csak a 40%-on értékeltem ki, amelyeknek nincs tanító fázisa, így minden eredmény pontosan összehasonlítható.

Algoritmus	HAS adathalmaz			TSSB adathalmaz		
	F1 érték	súlyozott F1 érték	Covering	F1 érték	súlyozott F1 érték	Covering
CLaSP	***0.7402***	*0.7235*	***0.6459***	***0.8923***	***0.8890***	***0.8385***
Window	*0.7121*	*0.7462*	*0.6100*	*0.7066*	*0.7705*	*0.5962*
BinSeg	*0.6820*	*0.7385*	*0.6412*	*0.6664*	*0.6560*	*0.3663*
BottomUp	*0.6029*	*0.6940*	*0.6175*	*0.5772*	*0.6376*	*0.5225*
TCN autoencoder	*0.4425*	*0.3173*	*0.4592*	*0.4876*	*0.4233*	*0.4472*
PELT	*0.7287*	***0.7835***	*0.5759*	*0.6483*	*0.6401*	*0.4215*
CLaSP_Window_BinSeg	*0.5053*	*0.5538*	*0.6164*	*0.4158*	*0.4395*	*0.6764*
PELT_BinSeg_BottomUp	*0.4505*	*0.5335*	*0.5364*	*0.3951*	*0.4260*	*0.4670*
Window_BinSeg_BottomUp	*0.4423*	*0.5246*	*0.5955*	*0.3520*	*0.5492*	*0.5492*
Pre-clustering the time series	***0.7426***	***0.7504***	***0.6368***	***0.8923***	***0.8890***	***0.8385***

2. táblázat - Az egyes algoritmusok és ensemble modellek által elért eredmények táblázata.

Mint ahogy az a 2. táblázatból is kiolvasható, a különböző algoritmusok eltérő eredményeket értek el a különböző adathalmazokon és metrikákban. Ahogyan az látható mindkét adathalmazon a CLaSP és az általam készített algoritmus éri el a legmagasabb értékeket. Kivételt képez a HAS halmazon mért súlyozott F1 érték, amiben a PELT hozza a legjobb eredményt. Ez egyfelől amiatt történt, hogy a PELT egy nagyon erős idősor szegmentáló algoritmus, mely ráadásul az idősor hosszában lineáris számítási kapacitású, másfelől amiatt történt, hogy az optimalizálása során a maximális szegmens számot 7-ben határoztam meg a munkám kezdetén. Ez volt az egyetlen algoritmus, ahol ezt az értéket ilyen alacsonyra választottam. Emiatt a sima és súlyozott F1 érték esetén is lecsökken a fals pozitív predikciók száma, ami javítja az értéket. Mivel a PELT algoritmust az előklaszterező megoldásom során nem használtam, ez megmagyarázza azt is, miért lesz alacsonyabb ezesetben az F1 értékei. Annak az oka, hogy a TSSB adatokon a CLaSP és az általam készített előklaszterező algoritmus ugyanazt az eredményt produkálja az, hogy A TSSB teszt halmazában mindösszesen 30, tanító halmazában 45 idősor volt, amely adatmennyiség kevés volt ahhoz, hogy ne mindegyik klaszteren a CLaSP nyújtsa a legjobb teljesítményt a tanítás során. Ellenben a HAS esetén már nagyobb mennyiségű adattal tudtam dolgozni, így megmutatkozott a súlyozott F1 score esetén az, hogy különböző algoritmusok a különböző klasztereken eltérő pontosságot érnek el. A lefedettség metrika esetén azt tapasztaltam, hogy a CLaSP minden algoritmusnál jobb eredményt ér el. Fontosnak tartom azt is megjegyezni, hogy a TCN autoencoder teljesítménye eléggé kilóg negatív értelemben, ami amiatt van, hogy az enkóder-dekóder architektúra és a sok paraméter megfelelőbb optimalizálása kellett volna, amit a jövőben érdemes folytatni is.



5. ábra - A különböző algoritmusok különböző klasztereken elért súlyozott F1 értéke a HAS adathalmazból képzett tanító halmazon.

Végezetül érdemes azt megvizsgálni, hogy melyik klaszteren melyik algoritmus teljesít a legjobban, és az a klaszter milyen karakterisztikájú idősorokat tartalmaz. Az erről készült eredményeket mutatja be az 5. ábrán szereplő oszlop diagramm. Ahogyan az pedig a 3. és 4. klaszterekről készült ábrán is látható, különösen az idősor hosszának és az autokorrelációs együtthatójának kontextusában jól elkülönül három klaszter, a negyedik pedig a variációs koefficiens tekintetében kivehető. A rövid, de magas autokorrelációs idősorokon mind súlyozott F1, mind lefedettség tekintetében a CLaSP éri el a legjobb teljesítményt. Míg az ugyanúgy magas autokorrelációval rendelkező, de hosszú idősorokon mindkét metrikában a BottomUp a legpontosabb. Igaz azonban, hogy ezen a klaszteren három algoritmus is nagyon hasonló teljesítményt nyújt, így ez lehetséges, hogy csak az adatokban levő zaj következménye. A rövidebb és alacsony autokorrelációval rendelkező idősorokon súlyozott F1 értékben a Window, míg lefedettségben a CLaSP éri el a legjobb teljesítményt. Végül pedig a kiugró variációs koefficiensű adatokon szintén a CLaSP a legpontosabb mindkét szempontból. A kapott eredmények nem meglepőek, hiszen, mivel a CLaSP egy osztályozó-alapú megközelítést használ, emiatt erős mintázat-alapú szegmentálásban. Rövid időszakokon belül, ha erősen ismétlődő struktúrák vannak, akkor könnyen rátanul az algoritmus. Hosszú idősorokon előkerül a BottomUp globális struktúrákra jól reagáló képessége, hiszen fokozatosan tudja felépíteni a szegmentálást. Illetve a CLaSP esetén ilyen sok adatponton már feltehetőleg sok hamis pozitív predikció lesz, ami rontja a súlyozott F1 értéket. A magas variációs koefficiens azt jelenti, hogy a szórás nagy az átlaghoz képest, tehát változékony az idősor. Ebben a CLaSP osztályozási logikája szintén erős. Más algoritmusok hajlamosak túlszegmentálni, mert nem tudják eldönteni mi a jelentős váltáspont, és mi nem. Abban az esetben pedig, amikor alacsony az autokorreláció, az adatsor nagyon zajos és kaotikus lesz. Ezeknek a lokális változásoknak a detektálásában a Window módszerek az erősek, mivel nem kell nekik hosszú távú mintázatokat érzékelni. A CLaSP ugyanakkor jobb covering-et produkált, ami arra utal, hogy ő is jól teljesít hasonló adatokon csak valószínűleg több hamis pozitív pontot is detektált, emiatt alacsonyabb súlyozott F1-et ért el.

2.4 Összegzés és kitekintés

Mint ahogy azt én is bemutattam, az idősorok szegmentálása egy komplex és sok innovatív megoldást magában rejtő témakör, amely továbbra is kutatások központi témája lehet elsősorban amiatt, hogy sokféle területen alkalmazzák az így kinyert plusz tudást, legyen szó akár egészségügyi, pénzügyi, viselkedés monitorozási vagy egyéb területekről. Kutatásom középpontjában az idősor szegmentálás együttes tanulással történő megvalósítása szerepelt, melyet a jövőben tovább fogok finomítani. Elsősorban a hierarchikus klaszterezési eljárás az, amin a megfelelő klaszterszám meghatározása további optimalizálást és hangolást igényel, mint azt az elért eredmények is mutatják. Ezenkívül érdemes további energiát fektetni a TCN autoencoder felépítésére és optimalizálására, mivel egy innovatív mély neurális hálókra épülő architektúráról van szó. Végezetül úgy gondolom, hogy a legjobb eredményeket elérő megoldásommal szintén érdemes a továbbiakban foglalkozni. A célom egy olyan megközelítés kidolgozása, amely nem igényli, hogy minden egyes új adathalmaz esetén külön tanítást és klaszterezést végezzünk. Ehelyett szeretnék létrehozni egy előre betanított modellt, amelyet nagy mennyiségű idősoros adaton tanítok be előzetesen. Ehhez további előre szegmentált idősoros adatokra van szükség. A tanítás során meghatározom az egyes klaszterek legjellemzőbb tulajdonságait és paramétereit. Ennek köszönhetően, amikor új adathalmazt elemzünk, a rendszer képes lesz automatikusan, a korábban tanultak alapján kiválasztani a legmegfelelőbb szegmentáló algoritmust, anélkül, hogy újra végig kellene vinni a tanítási folyamatot. Mindehhez szükséges az idősorok klaszterezésének további vizsgálata, és további paraméterek meghatározása, amelyek alapján meghatározhatók az egyes modellek erősségei és gyengeségei.

Összeségében tehát megfogalmazható az, hogy sikerült egy olyan módszert kidolgoznom, amely képes jobb eredményt elérni state-of-the-art idősor szegmentáló megoldásoknál, kihasználva az egyes modellek erősségeit. Ez a módszer nem lenne lehetséges az adathalmazok megfelelő előfeldolgozása és az adatok ismerete nélkül, melyből látható az is, hogy adatelemzési problémák esetén különös tekintettel kell lenni az adathalmazok megismerésére és előfeldolgozására, hiszen ezáltal hasznos információk nyerhetők ki, amelyeket a modellek tanításánál is fel tudunk használni.

3. Irodalomjegyzék, és csatlakozó dokumentumok jegyzéke

A tanulmányozott irodalom jegyzéke:

[1] Katy, 2024. október 15., Time series segmentation, CodeX, https://medium.com/codex/time-series-segmentation-8d6c6b328711, 2025.04.19.

[2] Patrick Schäfer, Arik Ermshaus, and Ulf Leser. 2021. ClaSP - Time Series Segmentation. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia. ACM, New York, NY, USA, 10 pages. https://www2.informatik.hu-berlin.de/~schaefpa/clasp.pdf, 2025.04.19.

[3] Patrick Schäfer, Arik Ermshaus, and Ulf Leser, 2023, ClaSP: parameter-free time series segmentation, Data Mining and Knowledge Discovery, https://github.com/ermshaua/claspy, 2025.04.19.

[4] Hyangsuk Min and Jae-Gil Lee, 2023 március, Temporal Convolutional Network-Based Time-Series Segmentation, IEEE, https://ieeexplore.ieee.org/document/10066661, 2025.04.19.

[5] Arik Ermshaus, Patrick Schäfer and Ulf Leser, 2023, ClaSP: parameter-free time series segmentation, Data Mining and Knowledge Discovery, https://github.com/ermshaua/time-series-segmentation-benchmark, 2025.04.20.

[6] Arik Ermshaus and Patrick Schäfer and Anthony Bagnall and Thomas Guyet and Georgiana Ifrim and Vincent Lemaire and Ulf Leser and Colin Leverger and Simon Malinowski, 2023, Human Activity Segmentation Challenge @ ECML/PKDD'23, 8th Workshop on Advanced Analytics and Learning on Temporal Data, https://github.com/patrickzib/human_activity_segmentation_challenge, 2025.04.20.

[7] arvkevi, https://pypi.org/project/kneed/, 2025.04.20.

[8] scipy.cluster.hierarchy.inconsistent, https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.inconsistent.html, 2025.04.29.

[9] https://centre-borelli.github.io/ruptures-docs/user-guide/detection/bottomup/, 2025.04.29.

[10] Gachomo Dorcas Wambui, Gichuhi Anthony Waititu, Anthony Wanjoya. The Power of the Pruned Exact Linear Time(PELT) Test in Multiple Changepoint Detection. American Journal of Theoretical and Applied Statistics. Vol. 4, No. 6, 2015, pp. 581-586. doi: 10.11648/j.ajtas.20150406.30

[11] https://centre-borelli.github.io/ruptures-docs/user-guide/detection/binseg/, 2025.04.29.

[12] Gaugel/ Reichert.: PrecTime: A Deep Learning Architecture for Precise Time Series Segmentation in Industrial Manufacturing Operations, https://arxiv.org/pdf/2302.10182, 2025.04.29.

Csatlakozó egyéb elkészült dokumentációk / fájlok / stb. jegyzéke:

Féléves munkámat egy Github repositoriumban tároltam, amely az alábbi linken található: https://github.com/czotterbenedek/time-series-ensemble-segmentation