

Laboradatok paramétereinek keresztmetszeti elemzése

Paraméterek közötti függőségi struktúra feltárása

Készítette: Villám Roland

Konzulens: Hullám Gábor

Bevezetés

Az orvostudomány fejlődésének felgyorsulása megkérdőjelezhetetlenül összekapcsolódik a modern informatika fejlődésével. Ahogy a processzorok fejlődnek, képesek a legszámításigényesebb feladatokat is elvégezni ésszerű időn belül. A genetika, a gyógyszerészet, a DNS kutatáson kívül még számos területen hoztak áttörést a számítógépes algoritmusok a korábbi módszerek helyett. Az adatok feldolgozása is sokkal gyorsabbá vált. Laboratóriumi mérési adatok milliói állnak rendelkezésre. Ezek összessége különböző összefüggéseket rejt, amelyeket az adatok együttes feldolgozásával lehet kinyerni belőlük.

Kutatásom során laboratóriumi adatok paramétereinek összefüggését vizsgáltam keresztmetszeti adathalmazon. A mért paraméterek függőségi struktúráját szerettem volna feltárni, illetve vizualizálni, további kutatások számára előkészíteni. Nem volt ismert, hogy az adatok

keresztmetszeti léte lehetővé teszi-e a paraméterek struktúrájának feltárását, és ha igen, mennyire lesz pontos az. Különböző Python programozási könyvtárak lehetőségeit hasonlítottam össze, hogy a legpontosabb módszert tudjam alkalmazni.

Adathalmaz

A használt adathalmazt a Semmelweis Egyetem Központi Laboratóriuma bocsátotta rendelkezésemre. Valódi páciensek valódi mérési adatai szerepeltek benne. A páciensek személyes adatainak védelmében az adataikat irreverzibilisen anonimizált (személyhez nem társítható) módon kaptam meg. Egy páciensen egyszerre több paramétert teszteltek, ezek összességével dolgoztam.

Az adathalmazt és a rajta végzett számításokat egy tanszéki szerveren végeztem, így az adatbázis a szerveren maradt végig. Az adatbázis fájl .csv kiterjesztésű volt, az adattagok | szimbólummal voltak egymástól elválasztva. Az ilyen kiterjesztést egyszerű beolvasni.

Keresztmetszeti adathalmazzal dolgoztam. Ez azt jelenti, hogy az adatok aggregálva voltak számomra, ezen kívül a mérések eredményei diszkretizálva. Így egy elvégzett mérés eredménye nem az adott paraméternek megfelelő mértékegységben volt megadva, pl. mg/L, hanem csak úgy, hogy az eredmény a megfelelő paraméterhez tartozó referencia tartományon belülre vagy azon kívülre esett. Referencia tartományon belüli eredmény 1 (normál), azon kívüli 2 (abnormális) értékkel volt jelölve. Ha az aggregált időintervallumon belül egy paramétert nem mértek, ott annak az eredménye 0 volt.

Az exportálás pillanata a keresztmetszeti vonal, amelytől indulnak az egyes időintervallumok.

Az adathalmaz fejlécének tagjai:

- RowNumber: A sor száma index funkciót látott el. Nem használtam, az adatbeolvasásnál automatikusan hozzáadott, fejléc nélküli, nullától kezdődő indexszel dolgoztam.
- Patient ID: A páciens neve helyett szereplő azonosító. Nem visszafejthető a páciens neve, csupán azt a célt szolgálja, hogy az egy pácienshez tartozó adatok együtt kezelhetőek legyenek.

- Sex: A páciens neme. Értéke m/f (férfi/nő).
- Age: A páciens kora, évben. Értéke nincs alsó egészszázra „kerekítve”, két tizedes jegy pontosságig van kiírva.
- LastMeasurementDate: Az utolsó mérés dátuma, év-hónap-nap formátumban megadva.
- LastMeasurementTime: Az utolsó mérés időpontja óra:perc:mp formátumba.
- Interval:
Értékei:
 - 1_month
 - 3_months
 - 6_months
 - 24_months
 - last_measurement
 - all_measurements

Az értékek az adatok exportálásától számított különböző időintervallumok. Egy pácienshez 6 sor tartozott, a fent említett időtartományonként egy-egy. Egy sorban az adott intervallumon belül elvégzett mérések összessége szerepelt. Az all_measurements sor a páciens összes mérési eredményét tárolta, míg a last_measurement az utolsóét.

A következő 3 fejléc esetén az XYZ a mért paramétereket helyettesíti. Például a CRP-t, amit a vérből lehet kimutatni, és megnövekszik a szintje gyulladás esetén.

- XYZ_successful: Az adott intervallumon belül elvégzett mérések száma az adott paraméteren.
- XYZ_prob_of_abnormal: Az adott intervallumon belül elvégzett mérések alatt az abnormális eredmények aránya. Ha pl. az intervallum alatt 12 mérést végeztek, amiből 5 abnormális lett, akkor ehhez a paraméterhez tartozó prob_of_abnormal: 0,416666.
- XYZ_max

Lehetséges értékei:

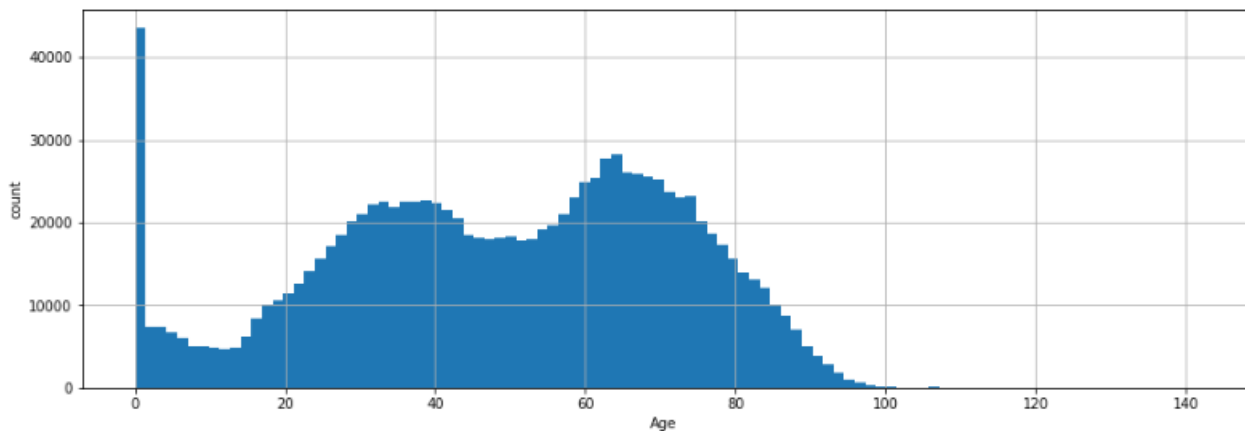
- 0: Az adott intervallumon belül nem mérték az XYZ paramétert.
- 1: Az adott intervallumon belül csak normál (referencia tartományon belüli) értékkel mérték XYZ paramétert.

- 2: Az adott intervallumon belül volt abnormalis (referencia tartományon kívüli) értéke az XYZ paraméternek.

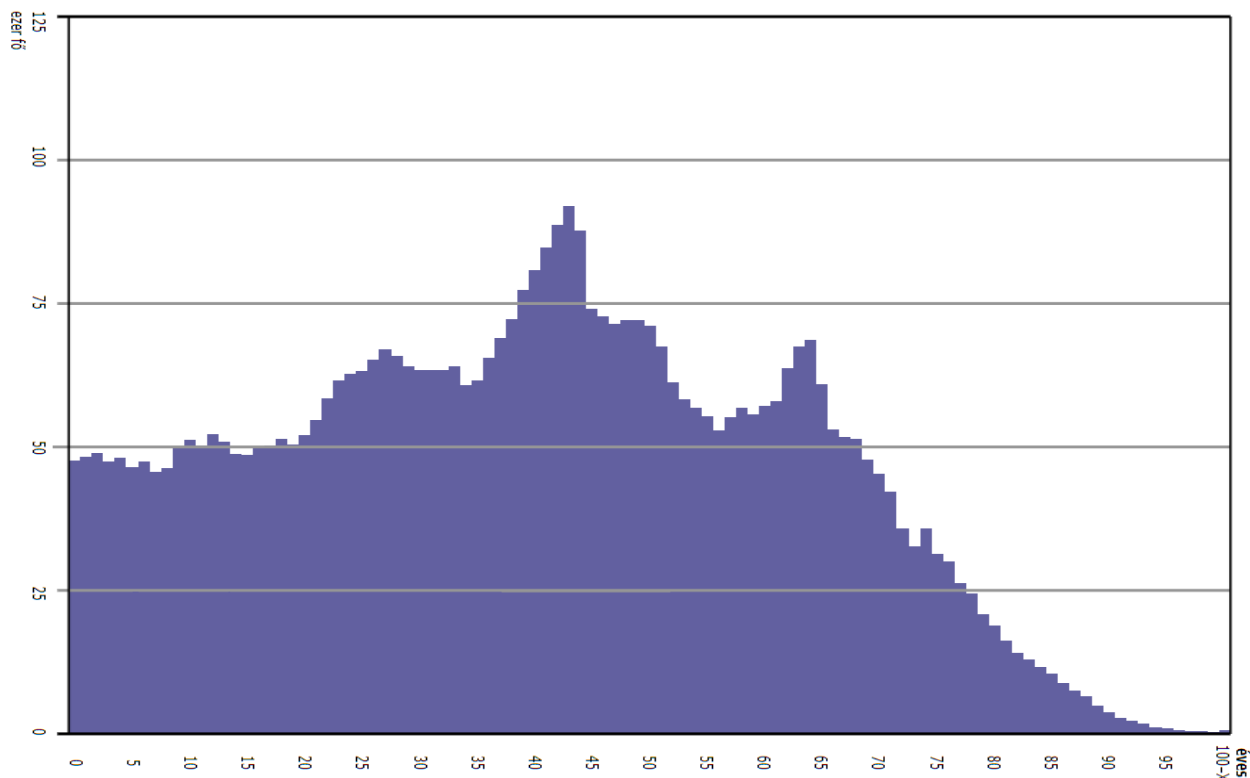
Könnyen belátható, hogy a keresztmetszeti adathalmaz néhány „információvesztést” tartalmaz. Egy időintervallumra tekintve ugyan ismerjük a mérések összességét, de azok egymáshoz viszonyulása számos esetben rejtve marad. Nem tudjuk, hogy például, ha egy páciensnél az elmúlt fél évben 2 paramétert 10-10-szer mértek, ezek teljesen, részben vagy külön-külön voltak mérve. Az adatok aggregációját egyszerűbben, sokkal kevésbé számításigényesen tudjuk kezelni, de mindenképp információvesztéssel.

Néhány adattag valamilyen okból kifolyólag hiányzott. Az adatbázisban ezekben a cellákban NaN (Not a Number) érték szerepelt. Ez történhetett valamilyen hiba miatt, illetve amiatt, hogy a privát páciensek neve is így jelent meg az adatbázisban. Az általam elvégzett folyamatokban figyelmet kellett fordítani az ilyen hiányzó értékekre és megszabadulni azoktól a soroktól, ami ilyen tartalmazott számomra releváns helyen.

Először egy tesztadatbázist kaptam meg, amelyen ellenőriztem, hogy az exportálás során nem lépett-e fel hiba. Ezen adathalmaz segített értelmezni, hogy a fent említett fejlécek pontosan mit jelentenek. 11 gyulladással kapcsolatban szokásosan mért paraméter szerepelt ebben az adatbázisban. Az adatok értelmezése után a páciensek kor szerinti eloszlását vizsgáltam. Ahogy az 1. ábrán látható, a legtöbb mérést 0-0,1 éves korban végezték. Belátható, hogy ezt a kiugrást az okozza, hogy az újszülöttektől születésük után, 48-72 órán belül kötelezően vesznek vért. Látható, hogy az életkor szerinti eloszlás bizonyos fokig illeszkedik a 2019-es magyarországi életkor szerinti eloszlásra (2. ábra). Észrevehető rajta a két kiugró gyakoriság 40 és 65 év körül.



Ábra 1: Páciensek kor szerinti eloszlása a teszt adatbázisban



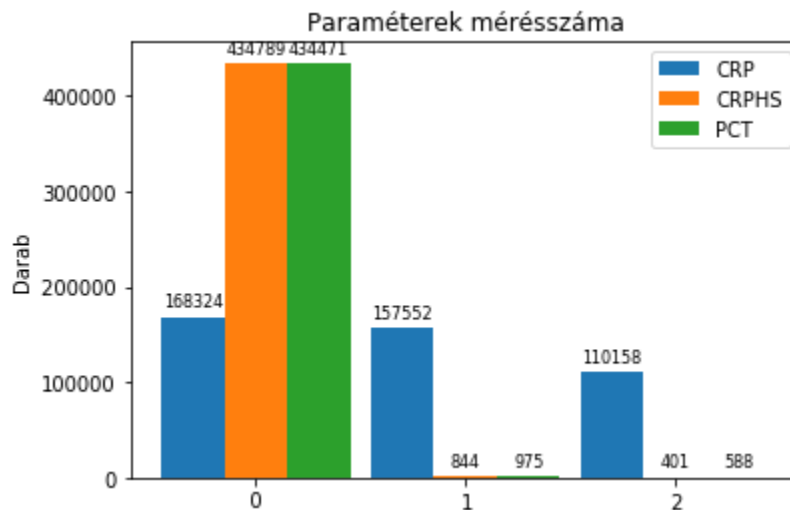
Ábra 2: Magyar lakosság életkor szerinti eloszlása 2019-ben

A páciensekhez tartozó `all_measurements` sorokat használva összehasonlítottam, hogy a tesztadathalmazon a különböző paraméterek milyen számban lettek mérve, illetve, ha mérve lettek, van-e eltérés, és melyik paraméter hányszor lesz normál vagy abnormális értékű. A 3. ábrán látható, hogy a CRP-t szinte ugyanannyiszor mérték normális vagy abnormális tartományok között (külön-külön), mint ahányszor nem mérték összességében. Ezzel szemben a CRPHS és a PCT paramétereket csupán összesen alig több, mint 1000 páciensnél mérték.

Ezen kívül a 10 leggyakrabban mért paramétert tartalmazó adathalmazzal dolgoztam. Itt az összes paramétert külön-külön legalább a páciensek 90%-ánál mérték. Feltételeztem, hogy ahol kevesebb olyan adat van, ahol nem mérték egy-egy paramétert, ott látványosabb lesz a paraméterek függőségi struktúrája.

Az előzőhöz hasonló, sokkal nagyobb adathalmaz a 80 leggyakrabban mért paramétert tartalmazó adatbázis. Ez körülbelül 3,3millió sort és 246 (6 alapadat és $3 \cdot 80$ paraméterhez tartozó) oszlopot tartalmazott. Ennek mérete 2,3 gigabyte volt, Pandas-ba való beolvasása is már eltartott átlagosan 2 percig.

Ennek egy részhalmaza volt a PSA specifikus paramétereket tartalmazó részadathalmaz, ami a PSA-t beleértve 13 paraméter együtt vizsgálatát jelenti.

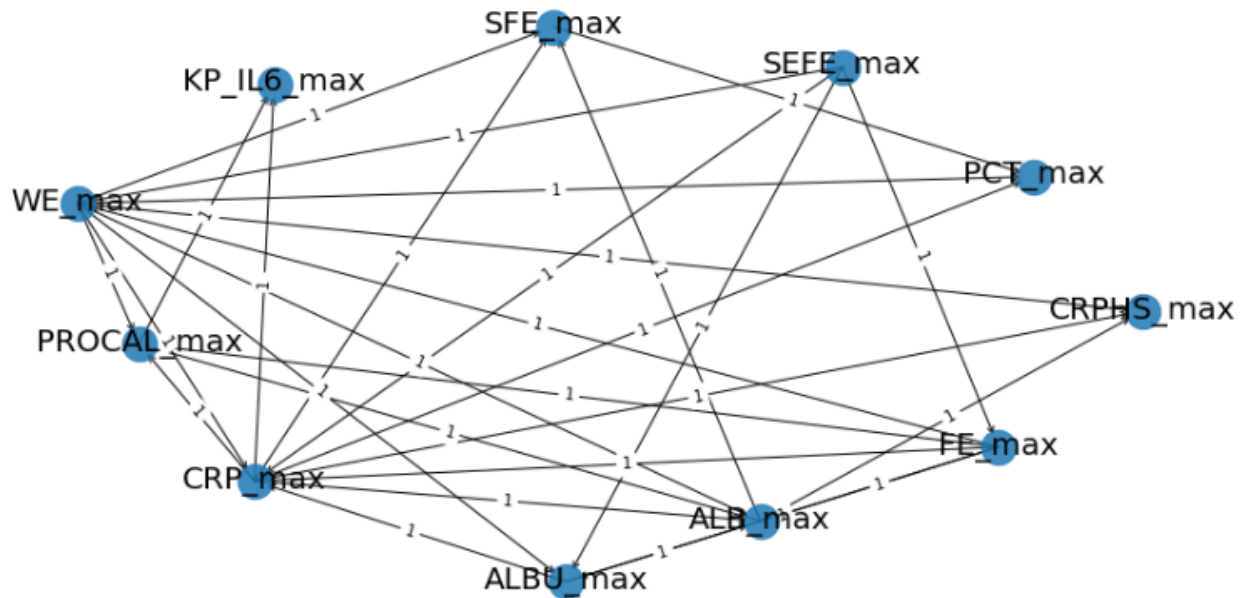


Ábra 3: Egyes paraméterek mérési aránya egymáshoz viszonyítva

Használt könyvtárak

Munkámat különböző Python könyvtárak segítségével oldottam meg. A legfontosabbak:

- Pandas [1]: Adathalmaz műveletek végezhetőek a segítségével. Adatbeolvasásnál, szűrésnél, módosításnál és fájlba írásnál használtam.
- Numpy [2]: Mátrix és tömb műveletek végezhetőek el vele egyszerűen. Gráfok szomszédsági mátrixainak összehasonításához használtam.
- Bnlearn [3]: Az alaptól R nyelvben létrehozott Bnlearn ihlette a Python-os verziót. Néhány struktúranulási algoritmus implementálva van benne. Elsőként ezt a könyvtárat próbáltam ki, és terveztem használni, de a lehetőségek ezzel korlátozottak voltak. A kirajzoláson kívül más funkciója nem volt, így a kapott struktúrát ennek segítségével szinte lehetetlen tovább elemezni. A 4. ábrán látható egy kirajzolás eredménye. Az iménti indokok miatt a későbbi eredményeimmel se hasonlítottam össze a Bnlearn algoritmus eredményeit.



Ábra 4: Bnlearn struktúra tanulási eredmény kirajzolva

- PGMPY [4]: A struktúratanuláshoz használtam ezt a könyvtárat, de más Bayes-hálóval kapcsolatos algoritmusokat is kínál, amelyeket használni lehet. Előnye, hogy könnyen használható; a részeredményeket és az eredményeket is több metódus segítségével újra lehet használni.
- Networkx [5], Graphviz [6]: Ezen könyvtárak segítségével tudom rendezett módon megjeleníteni a gráfokat, azok struktúráját fájlba írni vagy onnan kiolvasni.

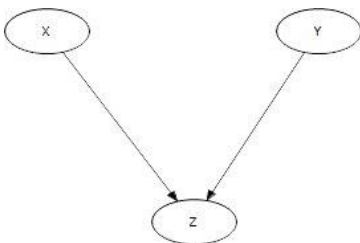
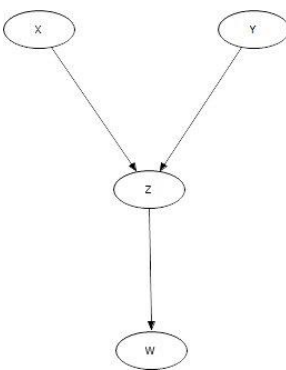
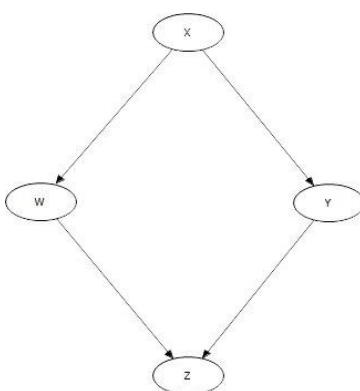
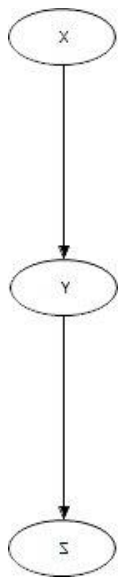
Struktúratanulási módszerek összehasonlítása

A PGMPY Python könyvtárból 5 különböző struktúratanulási módszert választottam ki összehasonlításra. Az exhaustive search számomra azért nem használható, mert működésének lényege, hogy az összes fajta struktúrát lepontozza, ami n paraméter esetén n^{n-1} struktúra kipróbálását jelenti, s ez 6 paraméter felett már nagyon sok időt venne igénybe.

Megmaradt PGMPY algoritmusok:

- Kényszer alapú megközelítés (Constraint based estimator)
- Pontszám alapú megközelítés, kereső algoritmussal, esetemben: Hegymászó keresés BIC, K2 vagy BD pontozással (HillClimb estimator BIC, K2, BD scoring)
- MMHC (Max-Min HillClimb)

A tanulás előtti legfontosabb lépés az volt, hogy kiválasszam az 5 lehetséges közül azt, amelyik a legjobban képes megtanulni a struktúrát az adatból, amelyik a legjobban leírja az adatot. Ehhez általam létrehozott struktúrához generáltam adathalmazt a BayesCube nevű programmal, amelyet a MIT tanszéken hoztak létre korábban. Segítségével meg lehet rajzolni a kívánt struktúrát, majd a paraméterek (Bayes-háló paraméterei) beállítása után tetszőleges méretű adathalmaz generálható hozzá. Különböző struktúrát leíró és különböző méretű adathalmazokat hoztam létre. A 5.ábra (táblázat) mutat képet a struktúráról, nevesíti a struktúrát és leírja a generált adatbázis méretét.

				
1 (a)	1 (b)	2	3	4
V-stuktúra		Y-struktúra	Gyémánt struktúra	Lánck struktúra
10000	10000	10000	10000	10000

5 (a)	5 (b)	5 (c)	6 (a)	6 (b)	6 (c)
Komplex struktúra 1			Komplex struktúra 2		
10000	20000	50000	10000	20000	50000

Ábra 5: Teszt struktúrák. 1.sor: struktúra képe, 2.sor: generált adathalmaz neve, 3.sor: struktúra neve 4.sor: generált adathalmaz mérete

Mivel 11 féle adathalmazon futtattam le 5 féle tanuló algoritmust, így mindenféleképpen automatizálnom kellett a kiértékelést, hiszen 55 különböző struktúrát kellett volna kézzel végig böngészni. A kiértékelést elvégeztem az eredmény irányított és irányítatlan gráf verzióján is. Az eredménygráfok hibáit számoltattam össze a következő kategóriák szerint:

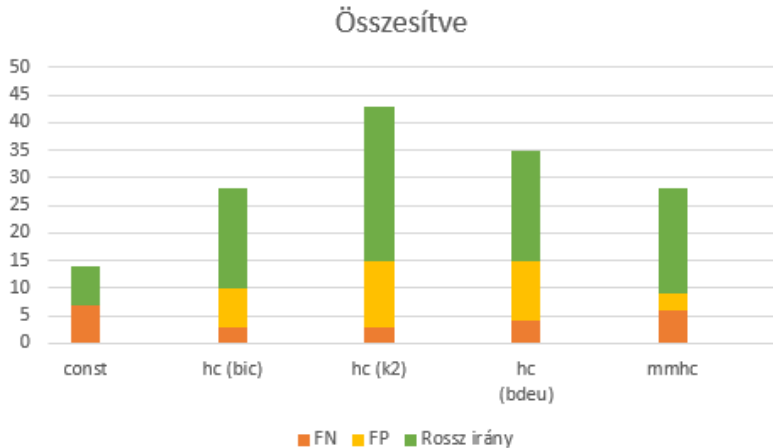
- Irányítatlan
 - FP (False Pozitív): van él, de nem kellene
 - FN (False Negatív): nincs él, pedig kellene
- Irányított
 - Rossz irányú él

A fent említett hibák megszámlálásához szomszédsági mátrixot használtam. A referencia struktúrákhoz sajnos kézzel kellett létrehoznom, de a tanult struktúrákhoz beépített módszer segítségével le tudtam generáltatni a szomszédsági mátrixot. A referenciából kivontam a tanult struktúra mátrixát, majd itt ez 1 és -1 értékeket összeszámolva megkaptam a fent említett hibakategóriák számait. Irányítatlan esetben a -1 FP-ot, az 1 FN-ot jelent. Irányított esetben elég volt csak az 1-eket számolni, majd kivonni belőle az irányítatlan FN darabszámát, vagy számolni a -1-eket, majd ebből kivonni az irányítatlan FP darabszámát, mindkettő megadja a rossz irányú élek számát a tanult struktúrában.

Az eredményeket a 6. és 7. ábrák (táblázatok) összesítik. A választásom a constraint based estimator, illetve a BIC pontozású HillClimb estimator-re esett. Összesítve is ezek érték el a legkevesebb hibát a tanulások során, de ami még fontosabb volt, hogy a komplexebb struktúrájú 50000-es adathalmazokon is ezek hibázták a legkevesebbet (ehhez hasonló, vagy nagyobb elemszámú adathalmazokhoz teszteltem, így ezeket nagyobb súllyal számoltam bele a döntésembe).

		10k	10k	10k	10k	10k	10k	20k	50k	10k	20k	50k
		1 (a)	1 (b)	2	3	4	5 (a)	5 (b)	5 (c) 50k	6 (a)	6 (b)	6 (c)
const:	FN:	0	1	0	0	0	1	0	0	2	2	1
	Irányítatlan: FP:	0	0	0	0	0	0	0	0	0	0	0
	Irányított: rossz irány:	0	1	0	1	2	0	0	0	1	1	1
hc (bic)	FN:	0	0	0	0	0	0	0	0	1	1	1
	Irányítatlan: FP:	0	0	0	0	0	1	3	0	0	2	1
	Irányított: rossz irány:	0	0	0	1	2	1	5	0	1	6	2
hc (k2)	FN:	0	0	0	0	0	0	0	0	1	1	1
	Irányítatlan: FP:	0	0	0	0	0	2	2	2	2	2	2
	Irányított: rossz irány:	0	0	0	1	0	4	4	4	5	5	5
hc (Bdeu)	FN:	0	1	0	0	0	0	0	0	1	1	1
	Irányítatlan: FP:	1	1	1	0	0	1	3	2	0	1	1
	Irányított: rossz irány:	2	1	2	0	0	1	4	4	1	2	3
MMHC	FN:	0	1	0	0	0	1	0	0	2	1	1
	Irányítatlan: FP:	0	0	0	0	0	1	1	0	1	0	0
	Irányított: rossz irány:	2	1	2	0	0	0	4	4	1	2	3

Ábra 7: Algoritmus tesztek részletes eredményei



Ábra 6: Metódusok összesített hibaszáma

Struktúratanulási módszerek működése

A struktúratanulás tulajdonképpen egy irányított aciklikus (körmentes) gráf (directed acyclic graph vagy DAG) struktúrájának tanulása az adatokból. Ennek két fő megközelítése létezik: a pontszám alapú (score-based) és a kényszer alapú (constraint-based). Esetünkben a két kiválasztott módszer pont egy-egy nagy kategóriába esik, ezek részletes bemutatása:

Pontszám alapú megközelítés

A pontszámon alapuló megközelítés [7] először egy kritériumot határoz meg annak értékelésére, hogy a struktúra mennyire illeszkedik az adatokra. Ezután a DAG-ok terében végez keresést, hogy megtalálja a legjobb pontszámmal rendelkezőt. Látható, hogy így a pontszám alapú keresést fel lehet osztani két részre: a pontszám metrika definiálására, illetve a keresési algoritmusra.

Pontszám metrika

Az általam használt pontozási módszer a Bayesian Information Criterion (BIC) [8]. Ennek képlete:

$BIC = k \ln(n) - 2 \ln(\hat{L})$, ahol:

- \hat{L} : Az M model MLE (maximum likelihood estimator) által maximalizált likelihood függvénye. Például: $\hat{L} = p(x | \hat{\theta}, M)$, ahol az $\hat{\theta}$ a paraméterek, amik maximalizálják a likelihood függvényt.
- x : A vizsgált adat.
- n : A minta mérete.
- k : A modell által becsült összefüggések száma, komplexitás.

A képlet első fele megbünteti a túl komplikált gráfstruktúrát és elkerüli a túlzott illeszkedést. A képlet második fele pedig jutalmazza azt, ha az adat jól illeszkedik a modellre. A képlet működésének így alapvető célja az, hogy olyan egyensúlyú struktúrát hozzanak létre, ami nem túl komplex és jól illeszkedik az adatra. Egy egyszerű modelltől indul ki, majd paramétereket ad hozzá, így a struktúra jobban illeszkedik az adatra, de egyre komplexebb is lesz, amíg eléri egy fordulópontot, ahol már a komplexitás felé billen az egyensúly.

Így tehát, ha a BIC metrikát használjuk, a kisebb pontszámú struktúrát preferáljuk. Fontos megjegyezni, hogy a BIC nem azt írja le, hogy egy modell milyen jól magyarázza az adathalmazt, milyen jól illeszkedik rá, hanem azt, hogy jobb egyensúlyban legyen a komplexitás és a leíró képessége a struktúrának az adatról, mint más modelleknek.

Kereső algoritmus

A hegymászó algoritmus egy optimalizációs eljárás, amely a lokális keresőalgoritmusok osztályába tartozik. Az eljárás egy kezdeti - véletlenszerű - megoldásból indul ki, majd iteratíván megkísérel egy mind jobb megoldást találni minden lépésben, mindig egy elemet megváltoztatva az eredményhalmazon, ameddig nem talál jobbat.

A hegymászó algoritmus csak a helyi szélsőértéket fogja megtalálni, amely nem feltétlenül a legjobb megoldás (a globális szélsőérték) az összes lehetséges megoldás közül. Ennek egy lehetséges feloldása az algoritmus kellő számú, többszöri futtatása véletlenszerű inicializálással.

További előnye, hogy a futtatás bármely pillanatában is szakítjuk meg a működését, a (rész)megoldás mindig elérhető.

Kényszer alapú megközelítés

A kényszer alapú megközelítés [9] szeparátor halmazokat, vagyis az általuk létrehozott feltételes függetlenségeket keresi. Ezekkel azonosítja a gráf élkényszereit, majd megkeresi a legjobb irányított körmentes gráfot, ami kielégíti a kényszereket. Például, megkülönböztetjük a V-, az elágazás- (fork) és láncszerkezeteket azáltal, hogy függetlenségi tesztet végzünk a két szélső paraméteren, változón, a középen lévő változó függvényében. Ez a legjobban akkor működik, ha vannak priori, szakértői ismereteink. Ezenkívül nagy adathalmazt igényel a tesztelési sikeresség érdekében, tehát kevésbé megbízható, ha a minták száma kicsi.

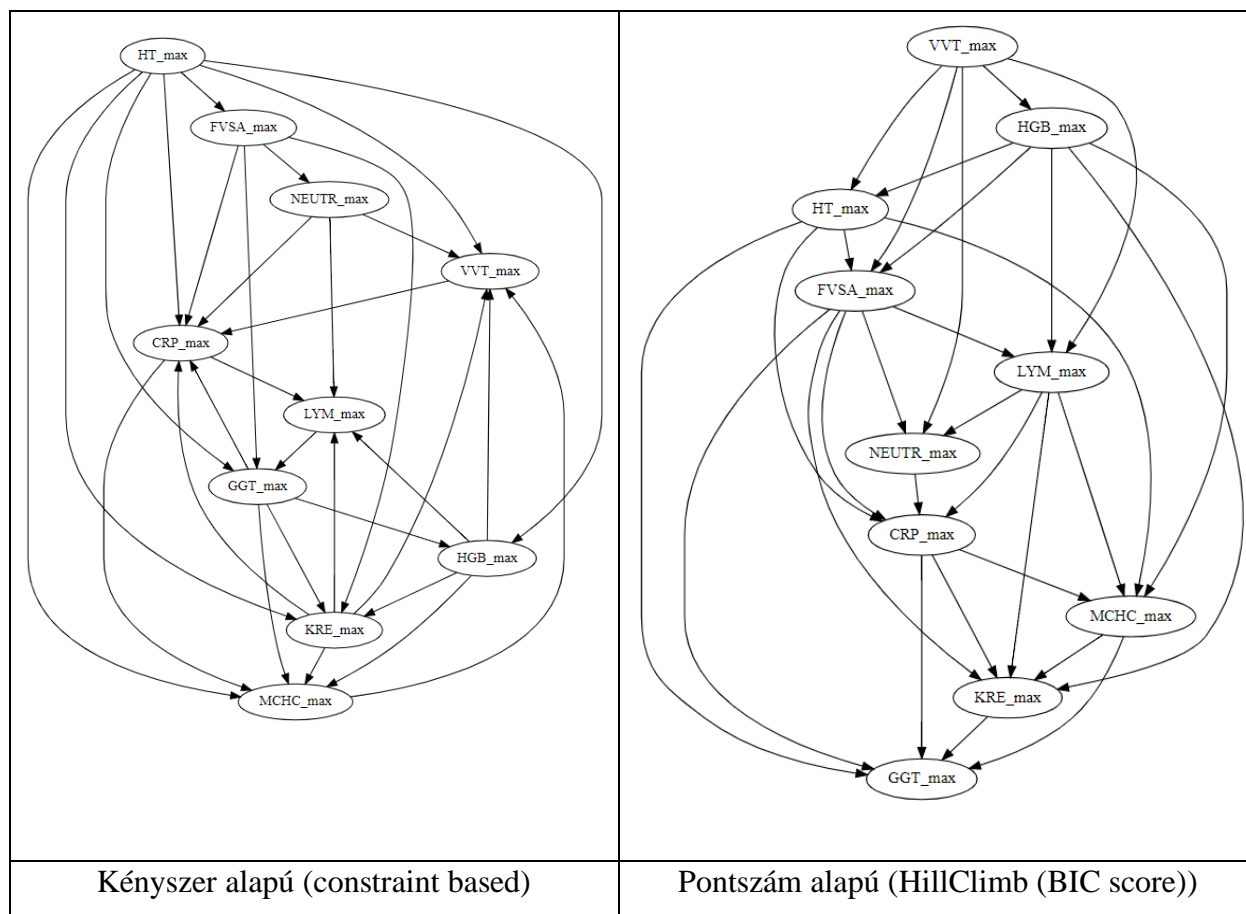
MMHC

A Max-Min HillClimb (MMHC) algoritmus [10] hatékonyan ötvözi a lokális tanulás ötleteit, kényszer alapú, valamint a keresési és pontozási technikákat. Először rekonstruálja egy struktúra skeleton-ját, majd egy HillClimbe-os keresést használ az élek pontosítására.

Eredmények

Top 10

A teszt adathalmazok és a gyulladásos paraméterek (amelyek szintén teszt jelleggel kerültek hozzám) után a top10 legtöbbet mért paraméteren futtattam le először a kiválasztott struktúranuló algoritmusokat. Páciensenként az all_measurements intervallumot használtam, így először erre szűrtem az adathalmaz sorait. Ezután a legfontosabb döntés az volt, hogy mennyire engedjem meg a 0 értékeket, azaz, ha valakinél valamilyen paramétert nem mértek. Ebben az adatbázisban könnyen megoldható lett volna, ha csak az olyan pácienseket tartom meg, akiknél mind a 10 paramétert mérték már, vagy beállítom, hogy bizonyos százalék felett legyenek mérve. Konzulensemmel viszont úgy döntöttünk, hogy egyáltalán nem érdemes egyetlen ilyen páciens adatait se eldobni az adatok közül, mivel az is információ, hogy egy paramétert nem mértek, ha nem valami hiba miatt, akkor direkt nem mérték azt a bizonyos markert. Így összesen körülbelül 530 000 sor rekordot használhattam fel a tanuláshoz. Az egyes algoritmusok külön-külön kb. 2-3 óra alatt futottak le.

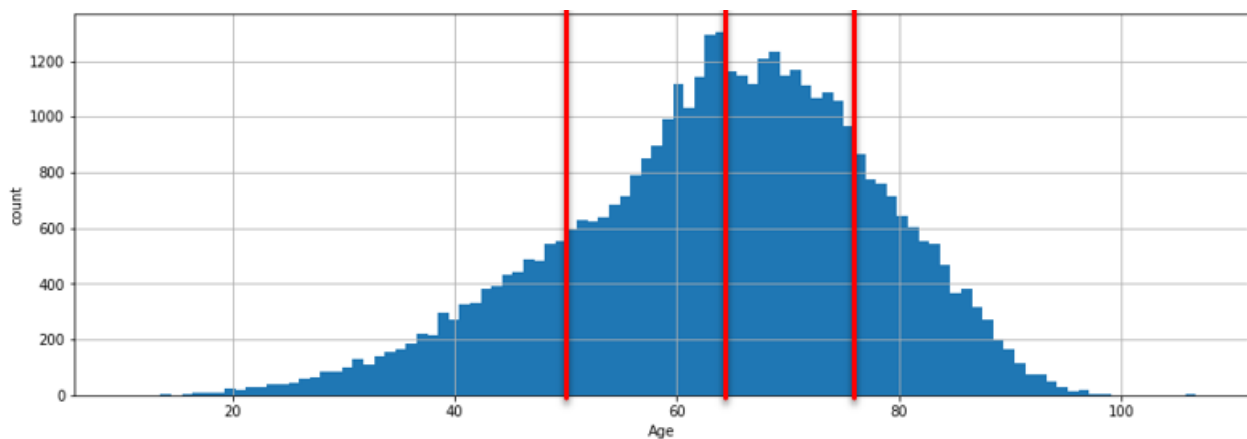


Ábra 8: Top10 adathalmaz eredmény struktúrái

PSA

A prosztata tumor specifikus adathalmaz 13 paraméteret foglal magába alapból (beleértve a PSA-t is). Ezeket a tényezőket rendelkezésemre bocsátották. A top10-es adathalmazhoz hasonlóan itt is az all_measurements intervallumokra szűrtem rá. Ezen kívül szűrtem arra, hogy a PSA paraméter mérve legyen, illetve, hogy a páciens neme férfi legyen. Egyes pácienseknél NaN érték szerepelt a nemüknél, de róluk nem szabad feltételezni, hogy férfiak, mivel voltak olyan női páciensek is nagyon kis százalékban, akiknél mérték a PSA paramétert. Ennek oka, hogy nőknél más betegségekre lehet specifikus a PSA normál értékétől való elváltozása. A szűrés utáni adatbázis kb. 47 000 sort tartalmazott.

Mivel minden használt számadat diszkrét volt az általam használt adatbázisban, ezért logikus ötlet volt diszkretizálni a páciensek korát is, mivel a Bayes-hálók tanulása esetén minden változó értéke vagy diszkrét vagy folytonos kell, hogy legyen, a vegyes változóhalmaz kezelése nehéz. A fent említett szűrés utáni (mért PSA érték, férfi) kor szerinti eloszlás a 9. ábrán látható. Az ember a leginkább 50-75 éves koráig van a leginkább veszélyeztetve a prosztatarák által. Emiatt két határvonalat ezekhez az életkorokhoz húztam be, illetve egyet 65 éves korhoz, hogy a görbe kb. egyenes része (65-75) egy tartományon belül legyen. Így a struktúratanulásba 14. paraméterként a kor (Age) is bekerült.

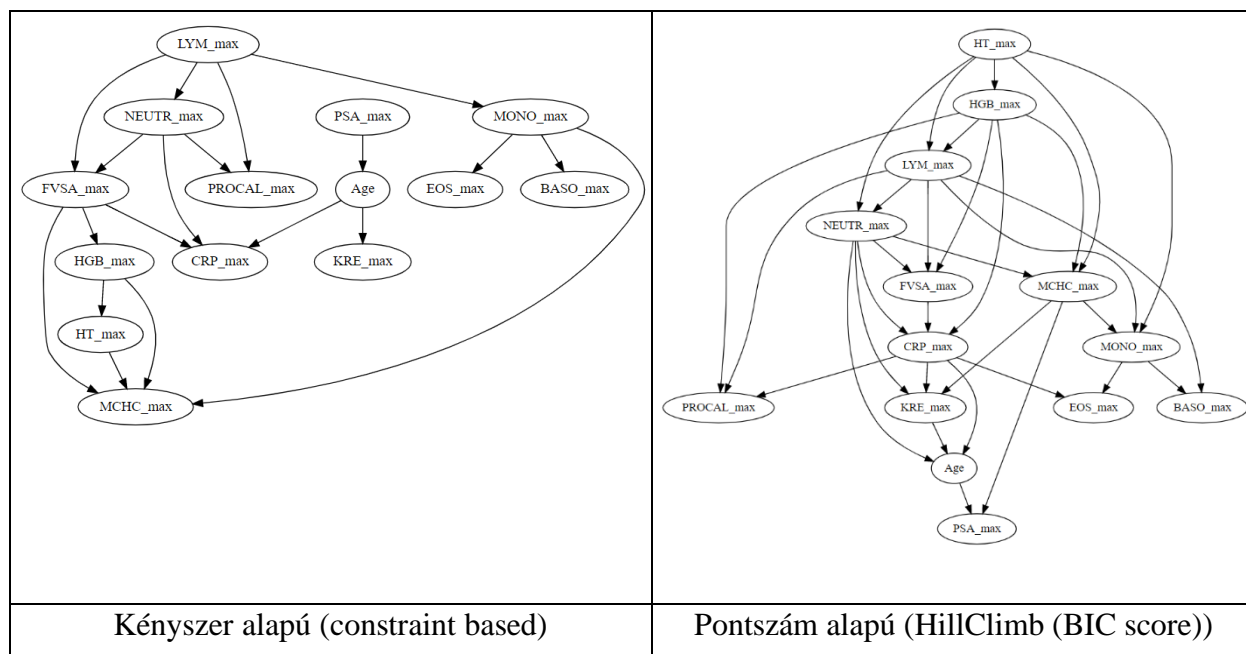


Ábra 9: PSA adathalmaz pácienseinek kor szerinti eloszlása, létrehozott korcsoportok

A két algoritmus eredményének (10.ábra) összehasonlításához referenciának a pontszám alapú algoritmus eredményét választottam (HillClimb (BIC)), mivel itt az alapvető cél az volt, hogy a

PSA markert befolyásoló paramétereket felfedezzem. Így ehhez képest a kényszer alapú algoritmus eredménye a már korábban említett FN, FP és a rossz irányú élek mellett most a TP (True Pozitív) éleket is számoltam (ott van él, olyan irányban, ahol kell).

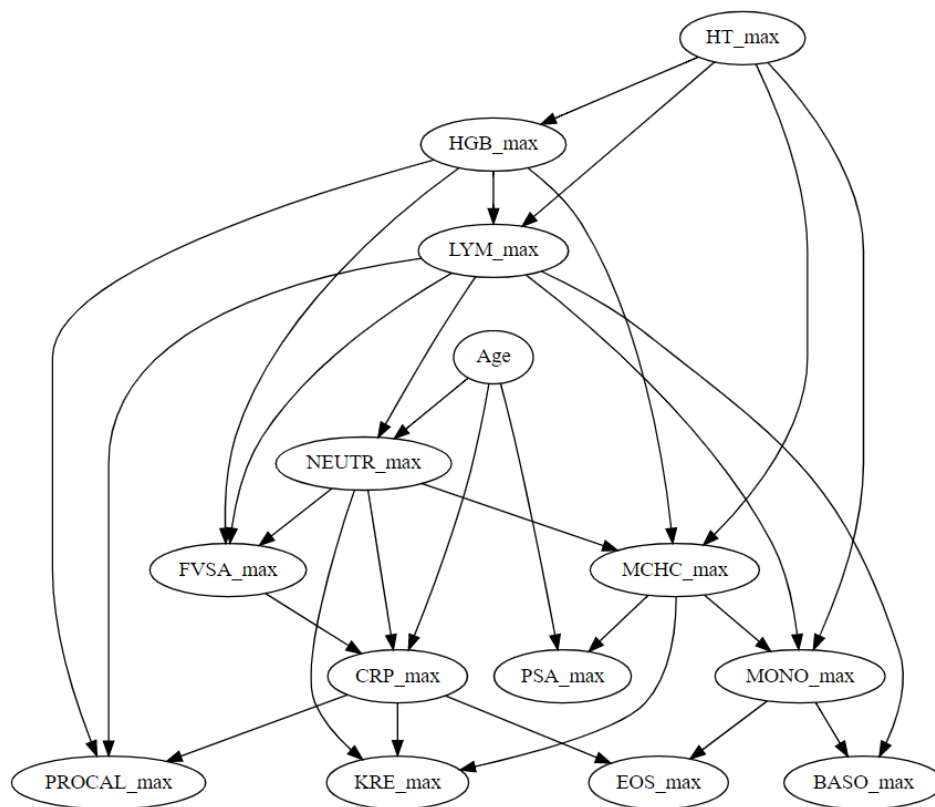
- TP: 11
- FP: 2
- FN: 15
- Rossz irány: 6



Ábra 10: PSA specifikus paramétereket tartalmazó adathalmaz eredménystruktúrái

Következő módosítás a fekete lista (blacklist) használata volt. Ezt csak a pontszám alapú algoritmusnál volt lehetőség beállítani. Segítségével ki lehetett zárni előre éleket a keresésből. Beállítottam, hogy a struktúra keresésben ne legyenek benne azok az élek, amelyek bármelyik paraméter felől a kor (Age) felé mutatnának, illetve azok, amelyek a PSA-ból mutatnának bármely másik paraméter felé. Így elértem, hogy a kor csak szülő paraméter legyen, míg a PSA csak leszármazott. Ennek eredménye a 11. ábrán látható.

Az eddig említett PSA-val kapcsolatos adathalmaz-tanulások maximum 1-1 órát vettek igénybe a tanszék szerverén.



Ábra 11: Feketelista implementálása utáni eredmény

További lehetőségek

A top80 leggyakrabban mért paramétert magába foglaló adatbázis tanítására nem volt elég 120 óra, így ennek az adathalmaznak a struktúratanulása az első jövőbeli feladat. A PGMPY további algoritmusokat, metódusokat kínál, amelyek elvégezhetőek a korábban említett adatbázisokon. A paraméter tanítás (Bayes-hálós paraméter), a csomópontok Markov takarójának kiíratása, további feldolgozása szintén jövőbeli lehetőség.

A struktúratanuláshoz a korábbi globális keresési algoritmusok helyett lokális keresésekből is lehetséges felépíteni a globális struktúrát. Emellett sztochasztikus struktúra tanuló algoritmusokat is be lehet vetni. Az MCMC, azaz Markov Chain Monte Carlo algoritmust lehetséges struktúratanulásra alkalmazni. Ilyenkor sztochasztikus keresést végzünk a DAG-ok terében.

Irodalomjegyzék

- [1] <https://pandas.pydata.org/>, Hozzáférés ideje: 2020.03.20.
- [2] <https://numpy.org/>, Hozzáférés ideje: 2020.03.20.
- [3] <https://erdogant.github.io/bnlearn/pages/html/index.html>, Hozzáférés ideje: 2020.03.20.
- [4] <https://pgmpy.org/>, Hozzáférés ideje: 2020.04.05.
- [5] <https://networkx.github.io/>, Hozzáférés ideje: 2020.03.20.
- [6] <https://graphviz.readthedocs.io/en/stable/manual.html>, Hozzáférés ideje: 2020.04.05.
- [7] [http://www.cs.technion.ac.il/~dang/books/Learning%20Bayesian%20Networks\(Neapolitan,%20Richard\).pdf](http://www.cs.technion.ac.il/~dang/books/Learning%20Bayesian%20Networks(Neapolitan,%20Richard).pdf)
- [8] https://en.wikipedia.org/wiki/Bayesian_information_criterion
- [9] <https://ermongroup.github.io/cs228-notes/learning/structure/>
- [10] <https://link.springer.com/content/pdf/10.1007/s10994-006-6889-7.pdf>