Production Systems and Information Engineering
Volume 11 (2), pp. 61-72 61
doi: 10.32968/psaie.2023.2.6

Nembináris Regressziós Fák vizsgálata

KOVÁCS LÁSZLÓ
University of Miskolc,
Hungary, Institute of Information Technology
kovacs@iit.uni-miskolc.hu

BÁN PÉTER
University of Miskolc,
Hungary, Institute of Information Technology
ban2@iit.uni-miskolc.hu

Absztrakt. A mesterséges intelligencia térnyerése egyre jelentősebb már a hétköznapokban is nem csak az iparban vagy orvoslásban. A hangsúly manapság a neurális hálókat használó metódusokon van viszont ez nem jelenti azt, hogy a korábban használt módszerek és algoritmusok elavultak lennének, továbbra is használatban vannak. A döntési fák, azon belül is a regressziós döntési fák hosszú múlttal rendelkeznek és ma már számos implementációjuk létezik. A regressziós fák többségében binárisak, amelynek oka hogy egyszerűek és nem vezet az adathalmaz töredezettségéhez. Hátránya viszont, hogy nagy és komplex fa épülhet ki amely rontja a modell pontosságát. Ezen cikk célja hogy megmutassuk, a nembináris regressziós fák képesek felvenni a versenyt a bináris felosztást végző társaikkal mint pontosságban mind sebességben.

Kulcsszavak: Regressziós fa, nem bináris, klaszterezés

1. Bevezetés

A mesterséges intelligencia és az adatbányászat egyre nagyobb szerepet játszik az életünkben. Gondolhatunk a személyre szabott hirdetésekre és ajánló rendszerekre, amelyekkel szinte mindenhol találkozhatunk. Évről évre jelentős fejlődés jellemzi a szakterületet különösen az utóbbi években. Gondoljunk csak az önvezetésre, a kép generálásra vagy a nyelvi modellekre. Ma már számtalan módszer és algoritmus létezik, többek között a döntési fák.

A döntési fát előszeretettel használják osztályozásra például az orvostudományban, viszont eredetileg regressziós feladatok megoldására találták ki.

A cikkben a nembináris regressziós döntési fák elemzését mutatjuk be, amelyben a legnagyobb hangsúly a pontosságon és a sebességen lesz, annak érdekében, hogy ezzel a módszerrel még hatékonyabban és gyorsabban lehessen dolgozni.

A döntési fák módszere talán elavult technológiának tűnhet, viszont gazdag 50 éves történelme [1] ellenére még mai napig elterjedt módszer, és nem csak azokon a területeken ahol közel a kezdetektől kezdve használjak, mint az orvostudományban, hanem az önvezetés során [2], az élelmiszeriparban [3], a megújuló energiaforrások növelésére [4], vagy a mérnökök fejlesztői munkáját segíti [5].

2. Irodalom

A regressziós döntési fák fogalmát 1963-ban vezették be az irodalomba, amely cikkben az Automatic Interaction Detection (AID) működését írták le és elemezték. [1] Az AID struktúra regressziós feladatokat képes megoldani. Működése során csomópontomként az adathalmazt megvizsgálva azt rekurzívan ketté osztja, még el nem ér egy bizonyos kritériumot és így kiépít egy hierarchikus fát. Az algoritmus az impurity mérőszámot használja annak megállapítására hogy mi az optimális felosztás egy adott csomópontban. A következő nagy előrelépés 1984-ben történt, amikor is publikálták az első cikket a CART algoritmusról [6] amely a Classification and regression tree rövidítése. Egy nagy újítása az algoritmusnak, hogy ahelyett, hogy valamilyen szigorú kritérium alapján leállna hamar a fa épülése engedi, hogy egy nagy méretű fát kiépítsen és utána visszametszi azt. Ez megoldja az alul illeszkedés és felül illeszkedés problémát, amivel az AID küzdött. Cserébe viszont több számítási kapacitást igényel. Továbbá 1986-ban egy kifejezetten osztályozásra alkalmas módszer is napvilágot látott amely az ID3 (Quinlan, J. Ross 1986) [7]. Az ID3 viszont nem az elvártaknak megfelelően működik bizonyos esetekben. Amennyiben egy attribútumhoz relatíven több, egymástól eltérő célérték tartozik, akkor hajlamos a felosztások során azt az attribútumot választani annak ellenére, hogy az nem a legmegfelelőbb. Ennek kiküszöbölésére továbbfejlesztették az ID3-at a C4.5 keretében (Quinlan, J. Ross 1993) [8] amely a gain ratio függvény segítségével normalizálja a kiszámított Infromation gain mérőszámot. A döntési fákat mind osztályozásra mind regressziós feladatokra használják.

Mára számos területen felismerték a döntési fákban rejlő lehetőségeket. Többek között az élelmiszeriparban. Az élelmiszeriparban fontos kérdés az, hogy a zöldséges és gyümölcsök frissen és jó állapotban kerüljenek el a fogyasztókig, hogy a lehető legkisebb legyen a pazarlás. Az áfonya jelentősége megnövekedett az utóbbi években viszont a szállítása és frissen tartása továbbra is kihívást jelent. Anastasia Ktenioudaki et al. azt szerették volna megállapítani, hogy hogyan őrizzék meg az áfonya minőségét addig amíg eljut a fogyasztóig. Ebben többek között fontos szerepet játszik a hőmérséklet és a páratartalom. Egy BRT modellt használtak, amely nagy pontossággal meg tudta jósolni az áfonya súly csökkenését, amely alapján óvintézkedéseket lehet bevezetni [3].

Bebizonyosodott továbbá, hogy a regressziós döntési fák az önvezető autók képességeit is képes kibővíteni. Az önvezetés technológia rohamosan fejlődik és egyre több autóban elérhető valamilyen korlátozással, mivel még vannak korlátait a technológiának. A kamerák „látási" viszonyait nagyban befolyásolja az időjárás, éppen ezért, hogy megállapítsák az út állapotát, radar szenzorok adatait használták fel egy kutatás folyamán és egy CART döntési fát építettek ki. Az esetek megközelítőleg 95% -ban pontos eredményt adott az út minőségéről [2].

3. Saját Módszer

Fentebb említett kutatások alapján láthattuk a döntési fák és azok között is a CART algoritmus előnyeit és hasznosságát, viszont fontos ismerni a hátrányait és hibáit. A CART annak érdekében, hogy felépítsen egy fát minden csomópontban elvégzi az optimális felosztást. Ebből az következik, minden csomópontban egy lokális optimumot keresünk. A lokális optimumok sorozatából viszont nem feltétlenül következik a globális optimum, amely a pontosság csökkenéséhez vezet. A bináris felosztás miatt továbbá egy nagy és komplex fa alakulhat ki, amelynek negatívumait például visszametszéssel lehet enyhíteni a problémát viszont nem oldja meg. Egy nembináris regressziós fa alkalmazásával csökkenthetjük a fa komplexitását és ezáltal gyorsabban pontosabb eredményhez juthatunk. Az ID3 esetén a nembináris felosztás adott, a CART esetén viszont nem egyértelmű a partíció optimalizálása. Javaslatunk a partíció optimalizálása egy HAC klaszterezésen alapuló módszert. Ennek indoka, hogy HAC módszer által, tetszőleges számú partíció összevonható és ezt az elvet szeretnénk alapul venni.

A HAC módszernek viszont egy nagy hátránya, hogy nagy számossággal bíró adathalmazok esetén rendkívül hosszú ideig tart az egyesítések elvégzése. Ezért, hogy ez ne befolyásolhassa az általunk javasolt módszer hatékonyságát, szükséges az adathalmaz elő feldolgozása a felosztások elvégzése előtt. Ezt a következő módon javasoljuk. Egy attribútumot kiválasztva a csomópontba beérkező adathalmazból, kiszámítjuk a célértékek súlyozott szórását a szomszédos elemek között, és ezt követően pedig azokat a szomszédos elemeket összevonjuk, amelyek az átlagos szórással megegyeznek vagy alacsonyabbak. Ezt a folyamatot addig folytatjuk még elérünk egy kívánt számosságot. Az előfeldolgozásnak köszönhetően a bejövő adathalmazt átalakítva, megfelelő hatékonysággal tud működni a nembináris felosztás módszere.

A nembináris felosztás megvalósítását a következő képpen javasoljuk. A tartományt több diszjunkt részre particionáljuk, a felosztás optimalizálásához pedig a HAC analógiáját követjük. A felosztási szám egy általunk tetszőlegesen választott paraméter, mely megadja, hogy egy felosztás során hány részre osszuk az adathalmazt. Kezdetben minden pont egy önálló partíciót alkot majd ezt követően azokat a szomszédos partíciókat vonjuk össze, amelyek az összevonást követően minimális költségnövekedéssel járnak. Ezt addig folytatjuk még el nem érjük a kívánt felosztási számot amelyet jelöljön F.

A költségnövekedés mértékének megállapítása függ attól, hogy milyen módszert használunk annak mérésére, azaz, hogy mit használunk hibafüggvényként. Egy regressziós döntési fa esetén a csomópontban lévő értékek átlagát vesszük alapul annak mérésére, hogy a modell az adott csomópontban mennyire pontos. A CART módszert alapul véve, a hiba mértékét az átlaghoz viszonyítva a Mean Squeared Error(MSE)-t használatával tudjuk mérni. Az általunk javasolt módszerben azt a kettő szomszédos klasztert vonjuk össze, amely az összevonásokat követően a legkisebb MSE érték növekedéssel rendelkezik.

A MSE-t az alábbi formában definiáljuk:

$$MSE(y) = \frac{1}{N} \times \sum_{i=1}^{N} (\hat{y} - y_i)^2 \quad , \; y_i \in y$$

Két szomszédos klaszter egyesítésének a súlyozott MSE értéke:

$$D_{i,j} = \frac{|k_i \cup k_j|}{|y|} * MSE(k_i \cup k_j)$$

Ahol $D_{i,j}$ két szomszédos egyesített klaszter súlyozott szórása, a k_i valamint k_j pedig egy-egy klasztert jelöl, az y pedig szülő csomópont célértékei

Ezt követően, amely két szomszédos klaszternek a legkisebb a $D_{i,j}$ értéke azokat egyesítjük egy klaszterbe. Ezután megint végig vizsgáljuk a klasztereket és ezt addig folytatjuk amíg el nem érjük az F számú klaszter számot. Végül pedig ez lesz a felosztás maga, a következő mélységi szinten.

Azt a felosztást választjuk amely legkisebb mértékben növeli a szórás mértékét. Ezt a következő képen számítjuk ki:

$$error = \sum_{i=1}^{F} (\frac{|y_i|}{|y|} * MSE(y_i)) - MSE(y)$$

Ahol y a szülő csomópont célértékei és y_i pedig az i-edik gyerek csomópont célértékei.

4. Implemetáció

Az implementálás során a Python programozási nyelvet alkalmaztunk. A programot jupyter notebookban írtuk meg és futtattuk. A lineáris modellek elkészítéséhez szintén a sckiti-learn Linear_regression függvényét alkalmaztuk. Az idő mérésére a time könyvtárat, valamint a számítások és tömbök kezelésére a math és numpy könyvtárakat vettük igénybe. A kód megírásához a python 3.11-es verzióját került felhasználásra.

A teszteléshez négy féle adathalmazt használtunk melyek mindegyike más-más szempont alapján került kiválasztásra.

5. Adatok

Blob

Az első adathalmaz, véletlenszerűen generált. Ennek megvalósítására a sckiti-learn, make_blobs metódusa került felhasználásra. Ez az metódus alapvetően klaszterezéshez való adatgenerálásra lett kitalálva . Az adat generálás során megadható, hogy mennyi

attribútummal rendelkezzenek a célértékek, valamint, hogy mennyi középpontot hozzon létre. Mindezen felül még a szórás mértékét is megadhatjuk.

A felhasznált adathalmaz 10.000 elemből áll, a központok száma 5, az attribútumok 4, valamint a szórás értéke 1.

Iris

A következő tanító halmaz az *iris*, ami az UCI oldalon legtöbbet letöltött adathalmaz. Ami a legfontosabb szempont, hogy a már létező nembináris felosztást használó algoritmusok nem működnek kis adathalmaz esetén, mert elfogynak az adatelemek még mielőtt egy megfelelő fa kiépülne. Ezáltal az *iris* adathalmaz megfelelő arra hogy megvizsgáljuk hogy az általunk javasolt módszer is rendelkezik ezen problémával.

Az adathalmaz 4 attribútummal rendelkezik. Ezen attribútumai valós értékeket vesznek fel. Az adathalmazt osztályozásra készítették el. A három osztályt, amivel rendelkezik lecseréltük számokra és így már alkalmas regresszióra. 150 tesztből épül fel.

Wine

Ezen adathalmaz 12 attribútummal rendelkezik, amelyek valós értékeket vesznek fel és 4898 tesztből épül fel. A leírás szerint az adathalmaz használható mind osztályozásra mind regresszióra. Továbbá mivel a többi adathalmazhoz képest lényegesen több attribútummal rendelkezik így láthatjuk, hogy képes-e kezelni ezen eseteket. Ezen adathalmaz is az UCI weboldalán található.

House

Ezt a tanító halmazt mi állítottuk össze és 10.000 elemet tartalmaz. Összesen 4 attribútummal rendelkezik és mindegyik attribútumhoz véletlenszerűen generáltunk értékeket a hozzájuk kitalált függvények segítségével. Az első attribútumhoz, amely a „méret” 0-tól 100-ig generálható véletlenszerűen egy értéket egy tizedesjegy pontosságig, majd ezt követően egy függvény segítségével állapítható meg hogy miként befolyásolja a célértéket, amelyet jelöljön P. A függvény a következő:

$$P_1 = \sqrt{méret} * 10 + \frac{0.3 * méret}{10}$$

A következő attribútum a „távolság”, melynek értéke 30-tól 100-ig tejed és egész értékeket vehet fel. A P-t a következő függvénnyel befolyásolja

$$P_2 = \frac{1000}{távolság} + 5$$

A harmadik attribútum, a „támogatás”, amely 0-tól 10-ig vehet fel egész értékeket. A P-t a következő függvénnyel befolyásolja:

$$P_3 = támogatás$$

A negyedik attribútum, a „keresett”, amely 3-tól 8-ig vehet fel egész értékeket. A P-t a következő függvénnyel befolyásolja:

$$P_4 = keresett^3$$

Ezen függvények együttesével egy nagy variációval rendelkező adathalmazt kapunk.

$$P = P_1 + P_2 + P_3 + P_4$$

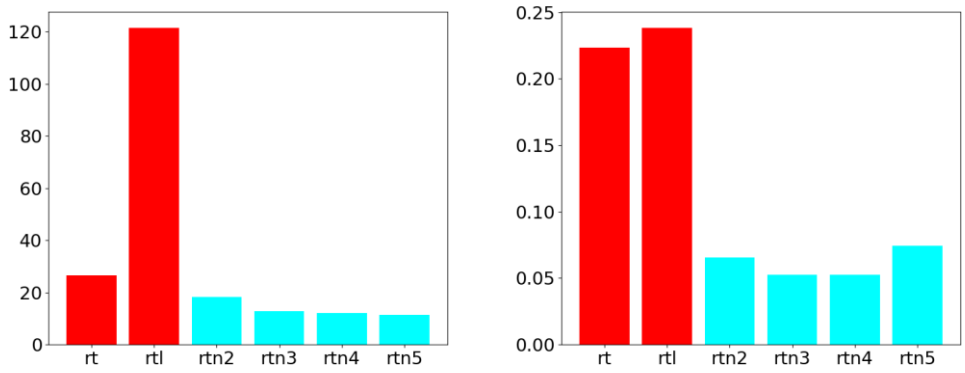
Az adathalmazok a beolvasását követően két részre lettek felosztva. Minden esetben a 80%-át használtuk fel tanításra és 20%-át pedig tesztelésre. Ezzel a módszerrel elkerülhető az, hogy azokon az adatokon történjen a tesztelés, amelyeket már ismer a modell. Ehhez a sckiti-learn sklearn.model_selection -ből a train_test_split függvényt alkalmaztuk.

A különböző módszerek, amelyek összehasonlításra kerültek:

Módszer	**Felosztás szám**	**Jelölés**
Regresszió fa	2	rt
Függvényosztály alapú Regressziós fa	2	rtl
Nem bináris Regresszió fa	2	rtn2
Nem bináris Regresszió fa	3	rtn3
Nem bináris Regresszió fa	4	rtn4
Nem bináris Regresszió fa	5	rtn5

6. Eredmények

6.1 Blob adathalmaz



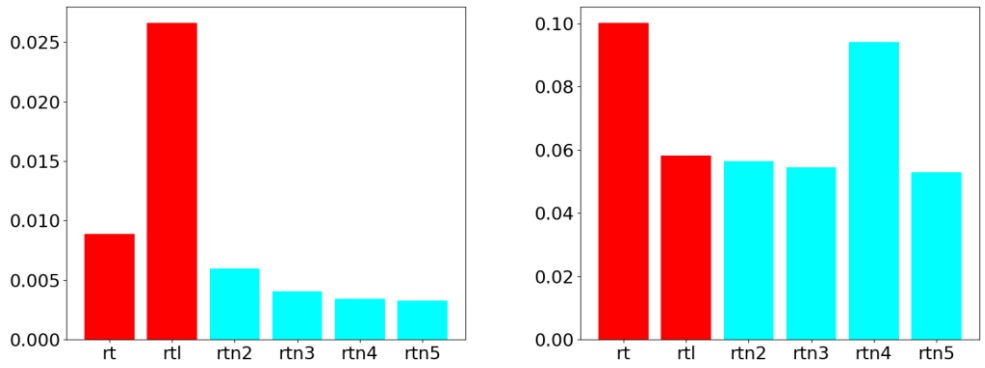
1.ábra. Blob adathalmaz futási ideje, valamint a hiba mértéke

Az első ábra jobb oldalán jól látható hogy tanítási idő tekintetében valamelyest gyorsabb volt a nembináris módszer. Az *rt* lényegesen gyorsabban végezte el a tanulást, mint az *rtl*. Ez nem is annyira meglepő mivel az *rtl*-nek még lineáris modelleket is létre kell hoznia így ez megnöveli mind a szükséges számítások számát mind az időt, ami alatt végez. Továbbá az is látható hogy ahogy növeljük a felosztás számot úgy csökken a tanítási idő.

Az első ábra bal oldalán található a hiba mértékét ábrázoló grafikon. Az rtn a többi módszerhez képest pontosabb volt. Mint láthatjuk a nembináris módszer alkalmazásával ezen adathalmazon nem csak hogy pontosabb eredményt érhetünk el, de gyorsabban is végez a döntési fa kiépítésével az algoritmus.

A *Blob* adathalmaz esetén az új módszer jól vizsgázott. Teljesítette az előre kitűzött célt, hogy nem csak gyorsabb legyen, de ne veszítsünk a pontosságból jelentős mértékben.

6.2 Iris adathalmaz



2.ábra. Iris adathalmaz futási ideje, valamint hiba mértéke

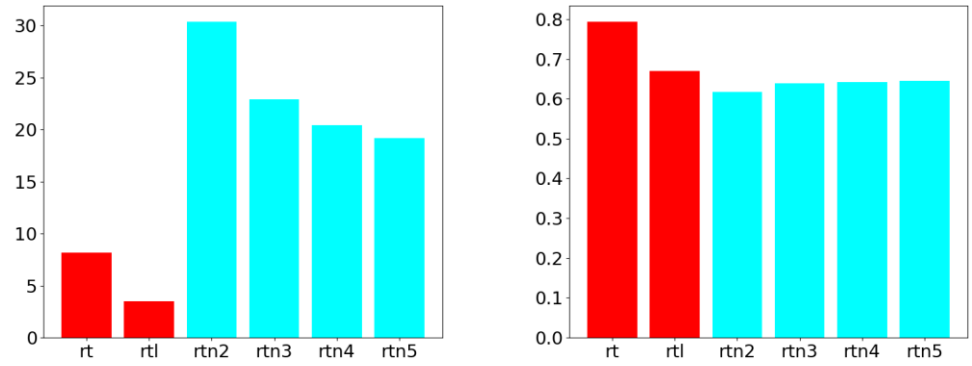
Az *iris* adathalmaz esetén fontos megjegyezni, hogy ez egy rendkívül kicsi adathalmaz a többihez képest. Ez nem probléma, hiszen, a nem bináris felosztás egyik kritikája hogy kis adathalmaz esetén nem hasznos, mivel azelőtt elfogynak a tanító halmaz elemei, mielőtt kiépülne egy jól hasznosítható fa.

A tanítási idő megint a *rtn esetében volt a legalacsonyabb. Itt is megfigyelhetjük hogy ahogy növeljük a felosztások számát úgy csökken a tanítási idő.*

Pontosság szempontjából láthatjuk, hogy a szimpla nem bináris felosztás hiába volt gyors viszonylag pontatlan. A lineáris modellt használó bináris felosztás ugyan lassabb volt cserébe viszont pontosabb is. A nembináris módszer alkalmazása esetén pedig egy esetet leszámítva minden esetben pontosabb eredményt kapunk az rt-nél valamint megközelíti a legjobban teljesítő *rtl* modellt.

Mivel a tanítási idő mindenképpen rendkívül rövid és a hiba mértékét tekintve egészen jól teljesített a modell, ezért láthatjuk hogy kis elemszámú adathalmaz esetén is képes kívánatos eredményt elérni.

6.3 Wine adathalmaz

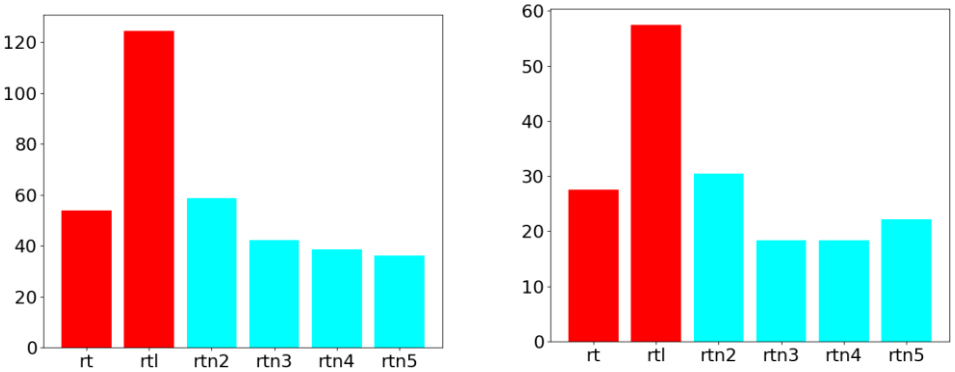


3.ábra. Wine adathalmaz futási ideje, valamint hiba mértéke

A *wine* adathalmaz esetén rendkívül érdekesen viselkedik a *rtln* és figyelemre méltóak az eredmények. Számítási időben, a nembináris felosztás lényegesen lassabb volt. Ez abból adódhat hogy az egyes attribútumokra történő felosztásnak a kiértékelése tovább tart mint a bináris fa esetén és itt ezt nem ellensúlyozza az hogy kevésbé mélyebb fa épül. A felosztási szám növelésével itt is csökken a tanítási idő.

Az általunk javasolt módszer megint pontosabb mint az a bináris felosztás viszont ebben az esetben az idő tekintetében az eddigiekhez képest az elvártnál lassabb volt. Látható hogy az lineáris modellt használó döntési fánál is pontosabb volt a nembináris felosztást használó módszer, annak ellenére hogy a bináris modell használata egy plusz előnyt ad a fának ahhoz hogy még pontosabb eredményt érjen el.

6.4 House adathalmaz



4.ábra. House adathalmaz futási ideje, valamint hiba mértéke

A futási idő ezen tanítóhalmaz használata esetén megint a legjobban teljesített. Mint látható a nembináris felosztás módszere gyorsabb mint a vele leginkább párhuzamba állítható már ismert módszer az *rt amennyiben az F nagyobb mint 2*. Továbbá a felosztási szám növelésével csökken a tanítási idő is, amely a fentebbi tanító halmazok esetén is láthattunk már.

A nembináris módszer pontosabb volt a mesterségesen létrehozott adathalmaz esetén is. A pontosság növekedése elsőre talán nem tűnhet soknak, viszont sem itt sem a fentebbi mérések esetén, viszont fontos megjegyezni hogy a nembináris felosztás ezen megvalósítása először történt meg amelyből látható, hogy igen is az esetek többségében mind a tanítási idő rövidebb mint a hiba mértéke kisebb.

7. Konklúzió

A döntési fa már több mint 50 éves történelme ellenére még mindig további lehetőségeket rejt a fejlődésre és mai napig hasznosnak bizonyult az élet különböző területein, mint az önvezetés, megújuló energia, élelmiszer ipar és az egészségügy. A fent bemutatott módszer fő szempontja a tanítási idő csökkentése a pontosság csökkenése nélkül, amely célt sikeresen teljesítette.

A nembináris felosztás valamely vizsgált variációja megmutatta hogy képes felvenni a versenyt az eddig használt módszerrel. Vagy időben vagy pontosságban vagy akár mind a kettőben kiemelkedő teljesítménnyel rendelkezett. Adathalmaz méretétől és komplexitásától függően láthatjuk az eredményeket, amelyek azt bizonyítják, hogy a nem bináris felosztás módszere nemcsak hogy gyorsabb, de akár pontosabb eredményt is elérhet. Továbbá láthattuk hogy a kritikák ellenére kis adathalmaz esetén is használható a modell.

Ezen eredményeket látva feltehetően, jövőbeli fejlesztésekkel tovább javíthatóak az eredmények.

Hivatkozások

[1] Loh, Wei-Yin. "Fifty years of classification and regression trees." International Statistical Review 82.3 (2014): 329-348. https://doi.org/10.1111/insr.12016

[2] Zhang, Chen, et al. "Road condition recognition in self-driving cars based on classification and regression tree." ICIC Express Letters, Part B: Applications 10.12 (2019): 1115-1122. https://doi.org/10.24507/icicelb.10.12.1115

[3] Ktenioudaki, Anastasia, et al. "Blueberry supply chain: Critical steps impacting fruit quality and application of a boosted regression tree model to predict weight loss." Postharvest Biology and Technology 179 (2021): 111590.s https://doi.org/10.1016/j.postharvbio.2021.111590

[4] AKINCI, T. Çetin, and H. Selçuk NOĞAY. "Application of decision tree methods for wind speed estimation." European Journal of Technique (EJT) 9.1 (2019): 74-83. https://doi.org/10.36222/ejt.558914

[5] Romeo, Luca, et al. "Machine learning-based design support system for the prediction of heterogeneous machine parameters in industry 4.0." Expert Systems with Applications 140 (2020):112869. https://doi.org/10.1016/j.eswa.2019.112869

[6] Breiman, Leo, et al. "Cart." Classification and regression trees (1984).

[7] Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1 (1986): 81-106.

[8] Quinlan, J. Ross. C4. 5: programs for machine learning. Elsevier, 2014.