Adatbányászat

Tardi Tamás, 2021 márc

A beadandómhoz az adathalmazt a openml-en találtam. A CSV több mint 8000 villámrandi adatát foglalja magában, ezeket 123 attribútummal jellemezve. Nem érzem célszerűnek az összeset leírni, inkább csoportokba szedtem őket. Hogy könnyebben értelmezhető legyen, nevezzük az egyik randizót A-nak a másikat B-nek, továbbá vannak többször is megjelenő tulajdonságok: vonzerő, intelligencia, őszinteség, humor, ambiciózusság, közös érdeklődés, ezekre már csak összefoglalóan tulajdonságokként fogok hivatkozni. A következő csoportok jelennek meg:

1. A és B alaptulajdonságai: kor, nem, foglalkozás, korkülönbség stb.
2. B preferenciái partnerével kapcsolatban a megadott tulajdonságokban
3. B értékeli A tulajdonságait
4. A preferenciái partnerével kapcsolatban a megadott tulajdonságokban
5. A értékeli saját tulajdonságait
6. A értékeli B tulajdonságait
7. Érdeklődés a felsorolt tevékenységek iránt (1-10 pont): sport, túrázás, olvasás, mozi stb., és ezeknek a korrelációja.
8. Randevúval kapcsolatos kérdések:
    a. A mennyire gondolja, hogy elégedett lesz az eseményen megismert emberekkel
    b. 20 ember közül A mit gondol hány partner fog érdeklődni iránta
    c. 20 ember közül A mit gondol hány darab lesz sikeres
    d. A mennyire kedvelte B-t
    e. A szerint mennyire valószínű, hogy B kedveli
    f. A és B ismeri-e egymást korábbról
    g. A döntése a randevú végén
    h. B döntése a randevú végén
    i. Sikeres volt-e a randevú

Osztályozás:

Számomra egyértelmű volt, hogy a találkozás sikerességét érdemes megbecsülni. Ennek érdekében a match oszlopot először numerikusból binomiálissá konvertáltam, majd label-nek választottam, és kiszűrtem a decision és decision_o oszopokat. Ha ezeket meghagyom akkor az osztályozók közel 99%-os pontossággal be tudják sorolni még a test adathalmazt is. A dataset tartalmaz hiányzó adatokat is, ezek mind numerikusak, mivel ezek általában preferenciákról szólnak itt az átlagot használtam helyettük.

Szerettem volna több módszerrel is elvégezni ezt a becslést, ehhez először az adathalmazt 70-30 arányban szétválasztottam. Az első futtatás után rájöttem, hogy problémát okoz, hogy a match oszlop ~84%-ban hamis értéket tartalmaz. Ez azt jelentette, hogy ha változtatás nélkül futtattam le a folyamatomat, akkor az az összes módszer 84% körüli pontossággal találta el a match oszlopot, a legjobb ebben az esetben a logisztikus regresszió volt 85%-kal, a decision tree bizonyos paraméterekkel pedig az összesre false-t becsült, és így is sem maradt el sokkal a többitől. Ennek kiküszöbölésére a Sample-t használtam, ez random módon kiválaszt kétszer ezer adatot, és ezzel már reálisabb eredményeket kaptam.

Az eredmények pedig a következők:

Módszer	Tanuló adathalmaz	Teszt adathalmaz
Decision Tree	78.64%	73.33%
Random Forest	97.21%	79.33%
Naive Bayes	79.29%	75.67%
Logistic Regression	81.14%	78.33%

Tapasztalatok:

Decision Tree: Az alapértelmezett gain ratio-val maximum 68.33%-os pontosságot tudtam elérni, a gini_index ettől egy lényegesen jobb 73.33%-ot produkál. A többi paraméter állítása nem hozott jelentős előnyt, a maximális mélység 6 után semmit sem változtat a végeredményen, a confidence és a minimal gain értéke pedig az alapértelmezett értékkel működik a legjobban. Próbáltam egy alacsony maximum mélységet megadni, hogy megmutathassam a fát, viszont mivel sok olyan oszlop van, ahol 1-től 10-ig megjelenhetnek értékek, ezért nagyon szerteágazó lett legenerálva.

Random Forest: Mivel a Decision Tree-re hasonlít, így itt is ugyanazt lehet tapasztalni, a gini index magasan a legjobb, és a mélység is elveszti jelentőséget egy bizonyos idő után.

Naive Bayes: Az egyetlen állítható paraméter a laplace correction, ha ezt kivesszük akkor gyengébben teljesít.

Logistic Regression: Alapértelmezett beállításokkal a többi módszert alulmúlta, viszont az L_BFGS solver-el és standardizációval az élmezőnybe került. Érdekességként megnéztem a súlyokat, a 3 attribútum, ami legjobban hozzájárul a match-hez: A mennyire kedvelte B-t, B mennyire találta viccesnek A-t, és A mennyire találta vonzónak B-t.

Megjegyzés:

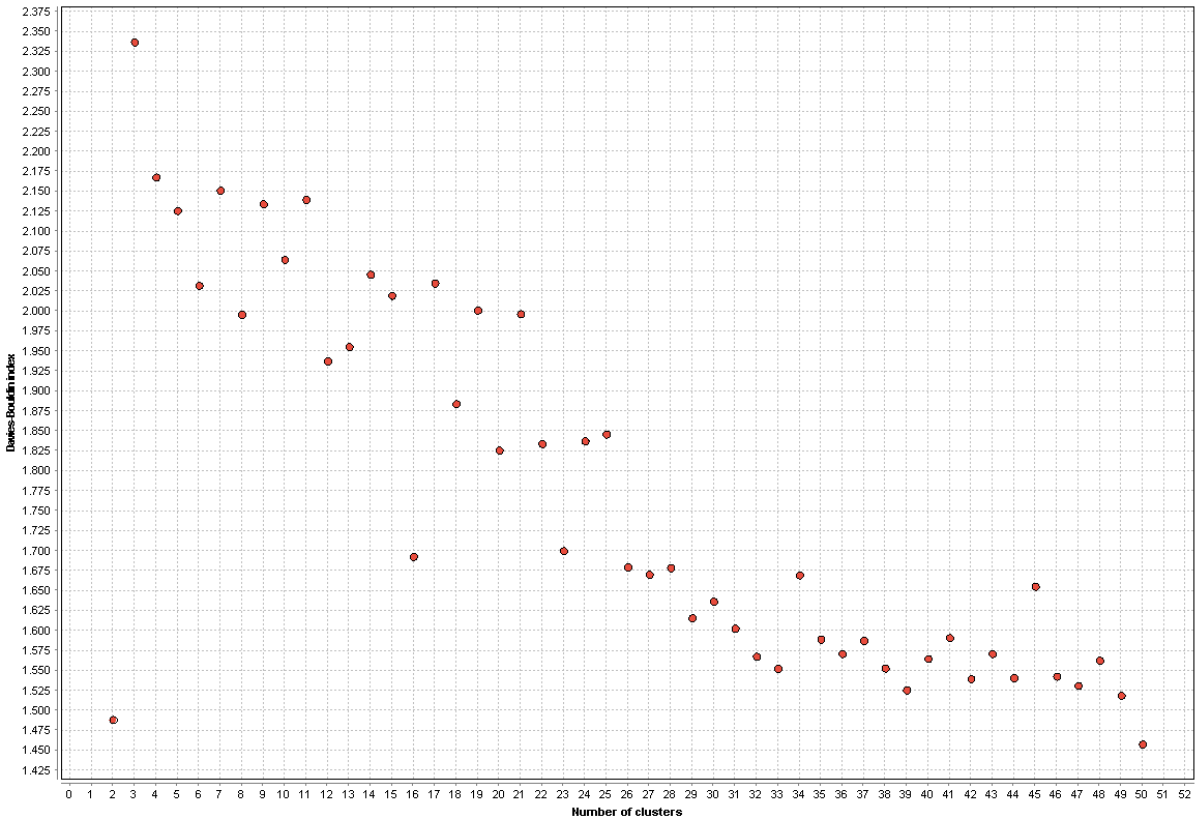
A RapidMinerben rengeteg kimenet volt vizsgálva, viszont hamar átláthatatlanná vált. Létrehoztam subprocesseket, hogy könnyebben lehessen értelmezni a folyamatot. Megjelent a preprocess for supervised subprocess, ami a bevezetésbe leírtak alapján előkészíti az adatokat és az {Módszer neve} Apply nevűek, ahol maga az 'Apply' van meghívva. Ezeknek csak a második kimenetét vezettem ki, ez a teszt adathalmaz teljesítményét mutatja meg.

Klaszterezés:

Az unsupervised learning részhez klaszterezést készítettem, azt szerettem volna megvizsgálni, hogy fel lehet-e osztani külön a nőket és a férfiakat különböző 'randizó típusokra', és ha igen, akkor hány klasztert célszerű készíteni.

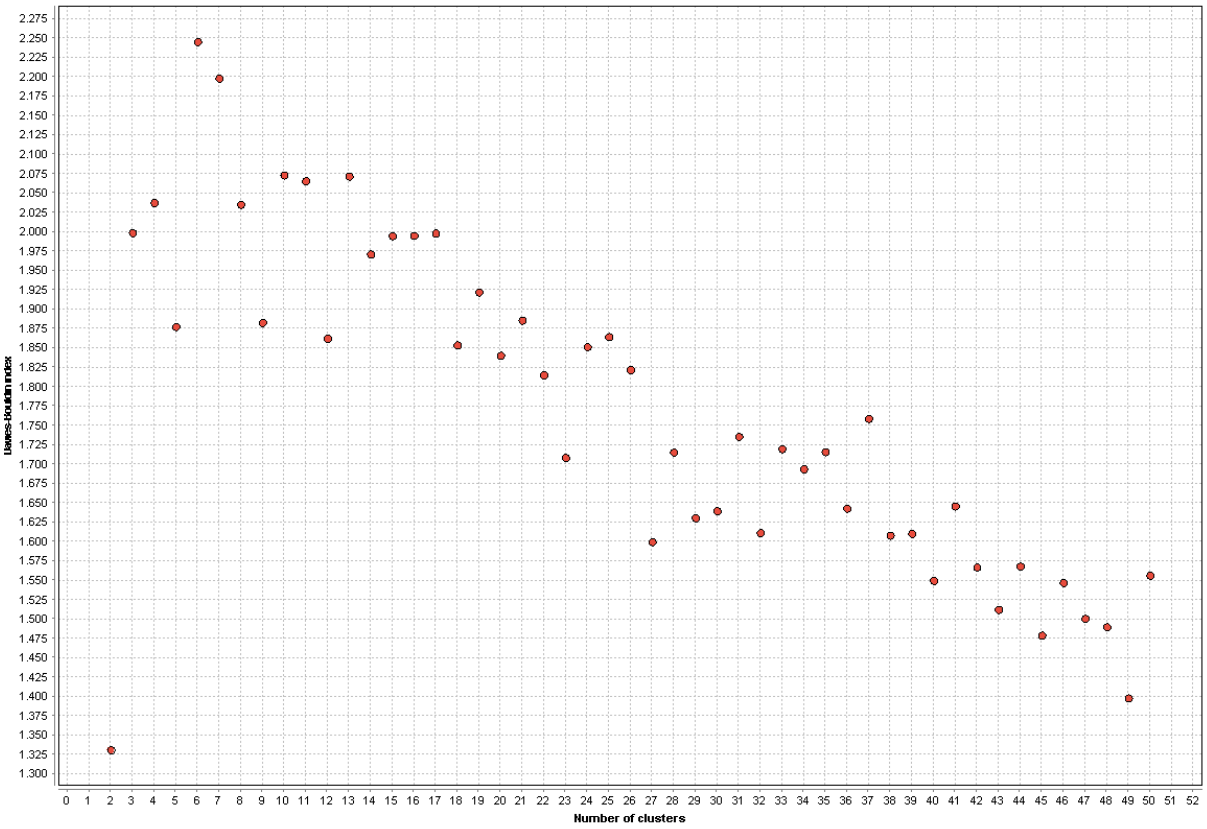
Az előzőhöz hasonlóan itt is szükség van előfeldolgozásra, mivel csak 'A' tulajdonságaira és preferenciáira vagyunk kíváncsiak.

A tanultak alapján egy loop-ot használtam az ideális 'k' megtalálására, a nőknél a Davies-Bouldin index kirajzolva így nézett ki:



Jól látható, hogy 2-nél és 16-nál van egy olyan lokális minimum, amit érdemes lehet használni, én a 16-ot választottam végül.

A férfiaknál ugyanez így néz ki:



Itt már egy fokkal nehezebb választani, nincs olyan kimagaslóan jó érték mint a nőknél. Hogy ne legyen túl sok klaszter, a 2, 5, 9, 12 és a 27 jöhet szóba, ezek közül a 12-t választottam.

Miután kiválasztottam a k értékeket már csak maga a klaszterezés maradt, erre a k-Means algoritmust használtam, a k érték kivételével alapértelmezett argumentumokkal.

Megjegyzés:

Ennél a feladatrésznél is létre lettek hozva subprocessek, egy az előfeldolgozásért felel, egy pedig magáért a klaszterezésért. A Loop Parameters is megnyitható, azt végül kivettem a folyamatból, mert kétszer is le kellett futnia, és ez jelentősen meghosszabbította a futásidőt.