Eötvös Loránd Tudományegyetem
Természettudományi Kar

Azonosságfeloldás adatbányászati módszerekkel

Szakdolgozat

Dobsa Dániel
Matematika BSc
Matematikai elemző szakirány

Témavezető: Lukács András
Számítógéptudományi Tanszék

Budapest
2017

Tartalomjegyzék

1. Bevezetés

Az adatbányászat az iparban, szolgáltatásokban egyre szélesebb körében ismert és alkalmazott módszer. Az azonosságfeloldás már kevésbé ismert, de számos helyen felbukkanó feladat. Dolgozatom célja egy azonosságfeloldási feladat bemutatása és megoldása adatbányászati eszközökkel támogatva. Az azonosságfeloldás feladata egy adatbázis rekordjainak kapcsolása, melyek különböző forrásból érkező adattáblákból is jöhetnek össze, egyértelmű kulcs nélkül. Ezt a problémát vezettem vissza a klasszifikáció feladatára. Az emberi agy meglepően hatékonyan képes eldönteni, hogy két, nem teljesen egyező cégnév ugyanarra az entitásra utal-e vagy sem. A cél egy olyan osztályozó létrehozása, mely jó teljesítménnyel tudja utánozni ezt a képességet.

Feladatom tehát két független forrásból érkező cégnevekről eldönteni, hogy van-e olyan cég, amelyik szerepel mindkettőben. A nehézséget az adja, hogy a klasszikus összekapcsoló módszerek csak teljes egyezés esetén adnak találatot, az adatunk viszont tartalmaz sok olyan esetet, ahol egy cég két különböző néven szerepel a két adatbázisban. Ennek oka lehet például elgépelés vagy cégforma-, és ez által cégnévváltoztatás. Ezért van szükségünk az azonosságfeloldás módszereire. A rekordkapcsolás kulcsszerepet játszik az üzleti intelligencia területén, hiszen itt elengedhetetlen egy egységes adattárház a konzisztens riportok gyártásához. További példa a Sales területről, ahol egy külső forrásból érkező, pénzért vásárolt adatbázisból ki lehet szűrni azokat a cégeket, akikkel már szerzősése van az adott vállalatnak, így időt és költségeket lehet spórolni. Ugyanezen problémakörhöz tartozik, hogy például az Ebay-t böngészve mi alapján ajánl nekünk az oldal hasonló termékeket. Ez is egy rekordkapcsolás, hasonló nevű és hasonló árú termékeket mutatva segít az oldal megtalálni a felhasználónak azt, amit keres.

A dolgozat bemutatja az azonosságfeloldás elméleti hátterét és a konkrét

feladat megoldását és kiértékelését. A 2. fejezet betekintést nyújt az azonosságfeloldás alapjaiba [1]. A 3. fejezetben a sztring alapú feloldás eszközeivel foglalkozunk, bemutatva az ismertebb sztringtávolságokat [2]. Majd áttérünk az adatbányászat alkalmazására [7][6], és a 4. fejezetben bemutatjuk a négy alapvető adatbányászati feladatot. Ennek részeként bemutatom azon osztályozó modelleket, melyeket alkalmaztam a rendelkezésünkre álló adattáblára [8][9], valamint az ismertebb modell teljesítmény mutatókat és ezek mérésének módszereit. Végül a konkrét feladat megoldása következik, az adatelőkészítéstől a modellek kiértékeléséig.

Ezúton szeretnék köszönetet mondani belső témavezetőmnek, Lukács Andrásnak, aki tanácsaival, észrevételeivel és szakértelmével nagyban hozzájárult a dolgozatom elkészítéséhez. Hálás vagyok továbbá Molnár Tamás külső konzulensemnek, aki a téma mély ismeretével gazdagította, tette értékesebbé a dolgozatot.

2. Azonosságfeloldás

Az azonosságfeloldás - angolul *entity resolution* - feladata a való világban előforduló, az adathalmazban különböző formában megjelenő, ugyanarra az entitásra utaló rekordok összekötése, ezen egy entitásra utaló rekordokból álló csoportok képzése. Például különböző módokon rögzített személynevek, apró eltérések egy cég nevében, vagy képek ugyanarról a tárgyról. Egy ideális világban persze rendelkezesünkre állna egy egyedi azonosító, ilyen például egy adószám, ami alapján lehetne kapcsolni az adatokat, de ez a való életben nem mindig van így. Nem túl meglepően magának az azonosságfeloldásnak is számos neve van, úgy mint deduplikáció, rekord kapcsolás, stb. Bár ezek, mint a fent említett feladatkör egy-egy hajszálnyit eltérő részfeladatait is jelentik. Deduplikáció alatt azt értjük, amikor egy adathalmazon belül csoportba rendezzük az ugyanahhoz a való világban lézető entitáshoz tartozó rekordokat, majd ezen csoporton belül megalkotjuk az entitást reprezentáló egy rekordot. Általában erre gondolunk, amikor azonosságfeloldásról beszélünk. A rekordkapcsolás feladata halványan eltér ettől, itt két különálló, már deduplikált adathalmaz között keressük a kapcsolatot. Ez feltételez egy már normalizált adatot, főleg relációs adatbázisokban. Alapvetően két csoportra lehet osztani a kapcsoló módszereket a kapcsolás megvalósításától függően:

- sztring alapú,
- link alapú.

A sztring alapú feloldás a rekordok nevében keres hasonlóságot és az alapján köti őket entitásokhoz, míg a link alapú az egyes rekordok kapcsolatai alapján teremti meg a kötést. A nagy közösségi oldalak tényleges felhasználószámuk eldöntésére együtt használják mindkettőt. Gondoljunk bele, két Molnár Péter néven futó profil teljesen különböző ismerősökkel nem tekinthető ugyanannak a személynek, viszont ha az ismerőseik nagyrészt megegyeznek, akkor

gyanús, és további szempontok szerint is lehet vizsgálni, mint például a profil aktivitása. Alkalmazási területek közé tartozik még - a teljesség igénye nélkül - a közegészségügy, spam detektálás, terrorelhárítás. Ebben a dolgozatban a feladatot a sztring alapú feloldás módszerével közelítjük meg, aminek alapvető eszközei a sztringtávolságok.

3. Sztring alapú feloldás

Számos matematikai definíció, algoritmus ismert, ami kiszámolja két sztring hasonlóságát, illetve távolságát. Ebben a fejezetben ezeket fogom bemutatni. Az 1. ábrán lévő példák szemléltetik, hogy a különböző, egymástól független forrásból érkező táblák összekapcsolásánál milyen sokféle feloldási problémával kell megbírkózni az adott attribútum formátumától függően. Az eltérésnek több oka lehet. Ezek közé tartozik az emberi hiba - más néven elgépelés - vagy automatizált rendszereknek köszönhetően egyszerűen más formátumban érkezik az adat. Ezért van szükségünk többféle sztringtávolság ismeretére és alkalmazására, hiszen mindegyiknek megvan a maga sajátossága, más tipusú 'hibákra' érzékenyek.

1.	John Blake	John Blaek
2.	John Blake	Jonathan Blake
3.	John Blake	Blake, John
4.	John Blake	Jon Blake
5.	John Blake	J. Blake

1. ábra. Személynév feloldás különböző forrásból érkező adatok esetén

3.1. Hamming-távolság

A Hamming-távolság a legegyszerűbb a sztringtávolságok közül. Ugyanolyan hosszúságú sztringek között a távolság megnézi, hogy a megfelelő poziciójú karakterek megegyeznek-e, a távolság pedig maga az eltérések száma lesz. Másképp fogalmazva, visszaadja azt a minimális számú karakterváltoztatást, amivel az egyik sztringből megkaphatjuk a másikat.

$$d_H(s,t) = \sum_{i=1}^{|s|} \delta(s_i, t_i), \qquad \text{feltéve, hogy } |s| = |t|.$$

Itt s_i és t_i rendre a két sztring i-edik eleme, $|s|$ pedig a karakterek száma, vagyis az s és a t sztring hossza, és $\delta(j,k)$ az úgynevezett *Kronecker-szimbólum*:

$$\delta(j,k) = \begin{cases} 1, & \text{ha } j = k \\ 0, & \text{egyébként.} \end{cases}$$

Abban az esetben ha nem egyforma hosszúak a sztringjeink, a módosított Hamming-távolságot használhatjuk. Legyen $|t| > |s|$, ekkor

$$d_{mH}(s,t) = d_H(s, t_{|s|}) + (|t| - |s|),$$

ahol $t_{|s|}$ a t szrting első $|s|$ karaktere, az összeg második tagja pedig a két sztring hosszának különbsége.

Például az *ANYA* és *ALMA* szavak Hamming-távolsága $d_H(s,t) = 2$, a *TERVEZÉS* és *TERMESZ* szavak módosított Hamming-távolsága $d_{mH}(s,t) = 4$.

3.2. Levenshtein-távolság

Az egyik legkorábbi és legismertebb szöveghasonlóság mérésére használt eszköz a Levenshtein-távolság. A távolság megadja azon megengedett szerkesztési műveletek minimális számát, amik ahhoz szükségesek, hogy egyik sztringet a másikba alakítsuk. A szerkesztési lépések számát egy egyszerű költségfüggvény határozza meg, ahol minden lépés költsége 1, kivéve a másolást, annak 0. Az alábbi műveletek állnak rendelkezésre:

- karakter másolása,
- karakter törlése,
- karakter beszúrása,
- karakter megváltoztatása.

Ezen szerkesztési műveletek segítségével definiálható a Levenshtesin-távolság:

$$d_L(i,j) = min \begin{cases} d_L(i-1,j-1) + d(s_i,t_j) & \text{karaktermásolás vagy -csere,} \\ d_L(i-1,j) + 1 & \text{karakter törlése,} \\ d_L(i,j-1) + 1 & \text{karakter beszúrása,} \end{cases}$$

ahol $d(i,j) = \delta(i,j)$). Valamint $d_L(i,0) = i$ és $d_L(0,j) = j$ minden $i \in [0,|s|]$ és $j \in [0,|t|]$ értékre definíció szerint. A Levenshtein-távolságnak több változta létezik, amelyek a $d(i,j)$ függvény részleteiben térnek el egymástól.

	A	N	Y	A
A	0	1	2	3
L	1	1	2	3
M	2	2	2	3
A	3	3	3	**2**

2. ábra. Levenshtein-távolság levezetése egy konkrét példán

Az 2. ábrán az ALMA és ANYA szavak távolságának kiszámítását láthatjuk. A táblázat alapján a két szó összes részsztringjének távolsága meghatározható. Tehát ez egy dinamikus programozási algoritmus, ami a balfelső sarokból tölti ki a mátrixot. Vegyük észre, hogy az azonos hosszúságú részsztringekhez tartozó részmátrixok szimmetrikusak. A két teljes sztring távolságát kiolvashatjuk a mátrix jobb alsó sarkából. Jelen esetben ez 2 (két

karakterváltoztatás), és a fő átló mentén jutottunk el a jobb alsó sarokba. Különböző alternetív szerkesztési utak is létezhetnek ugyanakkora költséggel.

3.3. Damerau-Levenshtein-féle távolság

A hagyományos Levenshtein féle szerkesztési távolság egy változata a *Damerau-Levenshtein*-féle távolság, ami annyiban különbözik az elődjétől, hogy egy további művelet is megengedett, a karaktercsere. Ez nem a hagyományos csere, ahol egy karaktert kicserélhetünk egy bármilyen másikra 1 költségen, hogy közelebb kerüljünk a másik sztringhez, hanem két szomszédos karakter cseréje az egyik sztringben ugyanilyen költségen, ugyanilyen megfontolásból. Damerau állítása szerint e négy szerkeztési művelet együttes alkalmazásával lefedhető az emberi elgépelésből származó hibák 80%-a [3].

3.4. OSA

Létezik egy úgynevezett megszorított szerkesztési távolság, másnéven OSA - *optimal string alignment*, ami a már ismert Damerau-Levenshtein-féle távolsághoz képest tartalmaz egy olyan megszorítást, hogy minden részsztringet csak egyszer lehet módosítani. Egy példa: legyen $s =$ "ba" és $t =$ "axb", ekkor

$$d_{DL}(s,t) = 2,$$

$$d_{OSA}(s,t) = 3.$$

Vegyük észre, hogy míg a Damerau-Levenshtein-féle távolság számításakor megtehettük, hogy az x törlése után megcseréljük a-t és b-t, úgy az OSA-nál már nem, hiszen így újra változtatnánk ugyanazon a részsztringen. Tehát például b és x törlése után egy b beszúrásával megkaphatjuk s-t.

3.5. Jaro-távolság

Ez a sztring távolság is a szerkesztési műveleteken alapszik, de a Levenshtein-félénél komplexebb. A lényege, hogy két sztring akkor hasonlít jobban egymáshoz, ha a rész betűsorozatai megfeleltethetők egymásnak. Rövid szövegekre lett kifejlesztve, főleg tulajdonnevekre. Hátránya, hogy matematikai értelemben nem tekinthető metrikának, mert nem teljesül rá a háromszög-egyenlőtlenség. Így azokra a módszerekre, amelyek kihasználják a távolságoknak ezt a tulajdonságát, nem használhatóak. A Jaro távolság képletében az $|S|$ és $|T|$ rendre S és T sztringek karakterhosszát jelölik, az m a megegyező betűk számát, míg c pedig a megfelelő helyen felcserélt betűk számát.

$$d_{JAR} = \frac{1}{3} \cdot \left(\frac{m}{|S|} + \frac{m}{|T|} + \frac{m-c}{m} \right)$$

Amennyiben az így számított távolság nagyobb, mint a hoszabbik sztring fele, úgy a két sztringet teljesen eltérőnek vesszük. Az algoritmust Winkler továbbfejlesztette, olyan megfontolásból, hogy a szavak elején a karaktersorrend helyessége fontosabb, mint a közepén vagy a végén. Ezért a sztringek közepén és végén lévő eltéréseket kisebb súllyal veszi figyelembe. Ebben a súlyozásban meg kell állapítani egy közös l hosszú prefixet, ahol:

$$l = \begin{cases} \text{közös prefix hossza}, & \text{ha } x \leq 4, \\ 4, & \text{egyébként}, \end{cases}$$

valamint egy $0 < p < 0,25$ skálázási paramétert, ami ajánlás szerint $p = 0,1$. Ekkor a módosított távolság képlete:

$$d_{JW} = d_{JAR} + l \cdot p(1 - d_{JAR}).$$

3.6. Jaccard index

A Jaccard index megadja két hatmaz metszetének és uniojának a hányadosát [6]. Lényegében tehát a két halmaz, és ezáltal a két sztring hasonlóságát.

$$J(S,T) = \frac{|S \cap T|}{|S \cup T|} = \frac{|S \cap T|}{|S| + |T| - |S \cap T|}$$

Amennyiben S és T halmaz is üres, úgy megegyzés szerint $J(S,T) = 1$. Így,

$$0 \leq J(S,T) \leq 1.$$

A Jaccard távolság, ami esetünkben két sztring különbözőségét méri, könnyen megkapható a Jaccard indexből, úgy, hogy kivonjuk 1-ből.

$$d_{JAC}(S,T) = 1 - J(S,T) = \frac{|S \cup T| - |S \cap T|}{|S \cup T|},$$
$$\text{ahol} \qquad 0 \leq d_{JAC}(S,T) \leq 1.$$

Ez a Jaccard indexből egyből adódik. Egy sztringből többféleképpen definiálhatunk halmazokat, amik aztán összehasonlításra kerülnek. Például a halmaz elemei lehetnek a sztring karakterei, vagy n egymás mellett lévő karakterből álló szeletei, tehát a sztring n-gramjai.

N-gram-nak nevezünk egy legalább n hosszú sztring n db karakterből álló részét [4]. Egy sztring n-gramjai alatt az összes így képezhető összefüggő szeletet értjük. Némely irodalomban találkozhatunk olyan definícióval, ahol a nem egybefüggű n elemű karaktersorozatot is n-gramnak hívnak (pl. egy sztring 1. és 3. karaktere is lehet a sztring egy 2-gramja, másnéven digramja), de mi most csak az egybefüggő szeletekkel foglalkozunk. Ezenfelül hozzáadunk még üres karaktereket az elejére és a végére, ami segít nekünk a szavak elején és végén lévő egyezések megtalálásában. Példaképp az ELTE szóból készített 3-grammok (trigrammok) így néznek ki:

`_EL, ELT, LTE, TE_, E__`

Vegyük észre, hogy az üres karakterek használatának egy másik előnye, hogy így minden k hosszú sztringből $k + 1$ n-gram képezhető, n-től függetlenül. Az első karakter elé egy üres karaktert teszünk, míg az utolsó mögé $n - 1$-et.

Az így kapott n-grammok halmazaira alkalmazhatjuk a Jaccard-indexet.

3.7. Leghoszabb közös szövegrész

Az angol neve - *Longest Common Substring* - ismertebb, magyar szakszövegekben is rendszeresen így utalnak rá. A módszer két vagy több sztring leghosszabb közös karaktersorozatát keresi. Mi most csak két sztring közös karaktersorozatának problémáját boncolgatjuk, az általánosítással nem foglalkozunk.

Adott két sztring, S és T rendre i és j hosszúsággal. Keressük azt a leghoszabb részsztringet ami megtalálható S-ben és T-ben is. A naiv megközelítés az, hogy hasonlítsük össze az összes S-ből képzett részsztringet az összes T-ből képzett részsztringgel, majd tetszés szerint eltároljuk a maximális hosszúságú részsztringet és/vagy a hosszt. Ennek azonban a futási ideje $\mathcal{O}(i^2 \cdot j^2)$ lenne, ami igencsak pazarló, annak fényében, hogy létezik egy dinamikus programozási algoritmus, ami jóval gyorsabb. Már találkozhattunk vele a Levenshtein távolságnál, ám most kicsit részletesebben is bemutatjuk a leghosszabb közös szövegrész példáján.

A dinamikus programozás az "oszd meg és uralkodj" elvén alapszik, tehát részfeladatok megoldásából építkezik alulról felfelé. Akkor alkalmazható, ha a részproblémák nem függetlenek, azaz közös részproblémáik vannak. Kell még, hogy teljesüljön a szuboptimalitás elve, ami azt jelenti, hogy az optimális megoldást adó struktúra megszorítása egy részfeladatra pont annak a részfeladatnak az optimális megoldását adja.

Vegyük tehát S és T leghosszabb közös szövegrészének dinamikus programozási megoldását:

$$LCS(S[1..i], T[1..j]) = \begin{cases} LCS(S[1..i-1], T[1..j-1]) + 1, & \text{ha } S[i] = T[j], \\ 0, & \text{egyébként.} \end{cases}$$

Lássuk az algoritmus működését. Legyen $S =$ APÁCA és $T =$ PÁCLÉ. Fontos, hogy nem szükséges az $i = j$ megkötés, különböző hosszúságú sztringekkel is működik az algoritmus. A táblázat jobb alsó sarkában kapjuk meg a megegyező részsztring karakterszámát, ami ebben az esetben 3.

	A	P	Á	C	A
P	0	1	1	1	1
Á	0	1	2	2	2
C	0	1	2	3	3
L	0	1	2	3	3
É	0	1	2	3	**3**

3. ábra. Leghoszabb közös szövegrész algoritmus levezetése egy konkrét példán

Kis változás az eddigi sztringtávolságokhoz képest, hogy itt minél nagyobb a szám, annál nagyobb a két sztring hasonlósága. Az algoritmus $\mathcal{O}(i \cdot j)$ időben lefut, valamint előnye még a szuboptimalitás, tehát egyszer kiszámolva megkapjuk S összes kezdőszeletének T összes kezdőszeletével vett hasonlóságát a mátrix megfelelő sorából és oszlopából kiolvasva.

Algorithm 1 LCS DP-megoldásának pszeudokódja [5].

```
function LCS(S[1 . . . i], T[1 . . . j])
    L := array(1 . . . i, 1 . . . j)
    z := 0
    ret := []
    for l = 1 to i do
        for k = 1 to j do
            if S[l] == T[k] then
                if l == 1 or k == 1 then
                    L[i, j] = 1;
                else
                    L[l, k] = L[l − 1, k − 1] + 1;
                    if L[l, k] > z then
                        z := L[l, k]
                        ret := S[l − z + 1..l]
                    else if L[l, k] = z then
                        ret := ret UNION S[l − z + 1..l]
                    end if
                end if
            else
                L[i, j] = 0
            end if
        end for
    end for
    return ret
end function
```

4. Adatbányászati módszerek

A legtöbb szakirodalom négy adatbányászati alapfeladatot határoz meg:

- **Előrejelzés:**
  Osztályozás és regresszió. Felismerési és előrejelzési feladatok, mint a beszédfelismerés, vagy annak eldöntése hogy egy gépjárműtulajdonos milyen valószínűséggel okoz majd balesetet. Itt megkülönböztetünk cél és magyarázó változókat. Célváltozó az, amit szeretnénk előrejelezni, tehát, hogy okoz-e balesetet a gépjárműtulajdonos vagy sem. Magyarázó változók pedig azok, amik alapján szeretnénk ezt megjósolni, például, hogy van-e gyereke, házas-e, stb. Mindkét fajta lehet kategorikus és folytonos is.

- **Klaszterezés:**
  Olyan osztályozási feladat esetében használjuk, amikor nem tudjuk előre az osztályokat, hanem pont ezek meghatározása a cél azzal, hogy csoportokba (klaszterekbe) soroljuk őket úgy, hogy az egy csoportokban lévő objektumok minél hasonlóbbak legyenek, míg a csoportok közöttiek minél különbözőbbek. Tipikusan ilyen feladat a piac szegmentálása, hasonló vásárlói csoportok meghatározása, irányított akciók tervezésére.

- **Minták és asszociáció**
  A gyakori minták keresésének feladatát legegyszerűbben a bevásárlóközpontos példával lehet jól bemutatni. Azt vizsgálja, hogy mik azok a termékek, amiket gyakran vásárolnak egyszerre a vevők. Ennek tanulmányozása lehetővé teszi például, hogy a bolt az egyik termék leértékelésével becsábítja a vásárlókat, míg a másik felemelésével kompenzálja ezt.

- **Anomália detekció**
  Eltéréselemzés, kiugró - angolul *Outlier*- pontok keresése. Olyan rekordok egy adattáblában, amelyek nagyban eltérnek az adatbázis általános jellemzőitől. Gyakran a hitelkártyacsalással, érvénytelen tranzakciókkal, hackertámadásokkal kapcsolatban merül fel az anomália detekció feladata.

Az osztályozás a legszélesebb körben elterjedt alapfeladata az adatbányászatnak. Elsőre talán furcsa lehet, hogy egy bizonyos a kéretlen levelek felismeréséhez használt algoritmus nem is különbözik annyira annak az eldöntésétől, hogy egy ügyfél befizeti-e a telefonszámláját. Mi is egy osztályozó algoritmust fogunk használni annak eldöntésére, hogy két cégnév ugyanahhoz a céghez tartozik-e.

4.1. Klasszifikációs feladat

A klasszifikáció, másnéven osztályozás az adatbányászati feladatok közül az előrejelzéshez sorolandó. Előrejelző módszerek között két nagy csoportot különböztetünk meg, a regressziót és az osztályozást. A különbség a két nagy csoport között, hogy míg a regresszió esetében az előrejelezendő vagy célváltozónkat intervallum skálán mérjük és nagyjából bármilyen értéket felvehet, a klasszifikációnál ez a változó diszkrét értékű. Például annak az eldöntése, hogy egy adott vásárló mennyit fog költeni a következő hónapban egy adott termékre, tipikusan egy regressziós feladat, míg azt, hogy két, látszatra különböző cégnév ugyanarra a cégre utal-e (igen/nem), klasszifikációnak hívjuk.

Adatbányászatban alkalmazott, ismertebb klasszifikáló módszerek:

- Legközelebbi szomszéd módszerek
- **Döntési fák, szabályok**

- **Random Forest**
- **Lineáris diszkriminancia analízis**
- Mesterséges neurális hálók
- Lineáris és logisztikus regresszió
- Naive Bayes és Bayes hálózatok
- SVM (Support Vector Machine)
- **Metaalgoritmusok** (Bagging, Boosting, stb.)

A vastaggal szedett módszereket fogom bemutatni, majd alkalmazni a rendelkezésünkre álló adaton.

4.2. LDA

A lináris diszkriminancia analízis célja olyan függvény létrehozása, amely a magyarázó változók lineáris kombinációját alkotva a célváltozók szerint a rekordokat a legjobban szétválasztja, diszkriminálja, majd ez alapján a megfigyelési egységeket valamelyik csoportba sorolja. Az eljárás osztálycímkéket használ fel, tehát felügyelt. A diszkriminanciaelemzés ideális esetben a következőket teszi fel:

- A célváltozók nominális skálán mérhető, tehát kategorikusak.
- A magyarázó változók metrikus skálán mérhetők, tehát folytonosak.
- Az összes megfigyelésnak függetlennek kell lennie egymástól.
- A csoportoknak egymást kizárónak kell lenniük, tehát egy megfigyelés a célváltozók egy és csak is egy értékét veheti fel.

- Csoportnagyságoknak nagyságrendileg egyező elemszámúnak kell lenniük.
- Homoszkedaszticitás teljesülése, miszerint adott magyarázó változó varianciájának a célváltozók különböző csoportjaiban hasonlónak kell lennie.
- A magyarázó változóknak normális eloszlásúnak kell lenniük.
- Magyarázó változók csak az adott célváltozóval függnek össze, egymással nem.

A gyakorlatban ezen elvi feltételek nem teljesülése nem feltétlenül eredményez rossz teljesítményű modellt.

Azt szeretnénk, hogy a tanítóhalmaz alapján minimális legyen a félreklasszifikálás, vagyis a különböző mintákat a legnagyobb valószínűséggel tudjuk a megfelelő osztályhoz rendelni. g osztály mellett egy adott X-ből származó és i osztályba tartozó x vektorra teljesüljön, hogy

$$P(i|x) > P(j|x), \qquad \forall j \neq i$$

Persze $P(i|x)$-et nem tudjuk megmondani, viszont ki tudjuk számolni a Bayes-tétellel:

$$P(i|x) = \frac{P(x|i)P(i)}{\sum_{\forall j} P(x|j)P(j)}$$

Ekkor a Bayes-szabály miatt, továbbra is i osztályba rendeljük a megfigyelést, ha:

$$\frac{P(x|i)P(i)}{\sum_{\forall k} P(x|k)P(k)} > \frac{P(x|j)P(j)}{\sum_{\forall k} P(x|k)P(k)}, \qquad \forall j \neq i.$$

A nevezők megegyeznek és pozítív előjelűek:

$$P(x|i)P(i) > P(x|j)P(j), \qquad \forall j \neq i.$$

Sajnos ez még mindig nem túl praktikus, mert sok osztály és sok dimenzió esetén $P(x|i)$ kiszámításához nagyon sok adat kell. Itt használjuk fel a magyarázó változók normális eloszlását:

$$P(x|i) = \left(\frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{C}_i^{\frac{1}{2}}|} \right) e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mu_i)}.$$

Ahol μ_i az i csoport átlagvektora, míg $\mathbf{C}_i$ a kovariancia mártixa. Ezt behelyettesítve a Bayes-formulába kapjuk:

$$\left(\frac{P_i}{(2\pi)^{\frac{n}{2}} |\mathbf{C}_i^{\frac{1}{2}}|} \right) e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mu_i)} > \left(\frac{P_j}{(2\pi)^{\frac{n}{2}} |\mathbf{C}_j^{\frac{1}{2}}|} \right) e^{-\frac{1}{2}(\mathbf{x}-\mu_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mu_j)}.$$

Mivel $(2\pi)^{\frac{n}{2}}$ mindkét oldalon megegyezik, ezért leoszthatunk vele:

$$\left(\frac{P_i}{|\mathbf{C}_i^{\frac{1}{2}}|} \right) e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mu_i)} > \left(\frac{P_j}{|\mathbf{C}_j^{\frac{1}{2}}|} \right) e^{-\frac{1}{2}(\mathbf{x}-\mu_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mu_j)}.$$

Mindkét oldal természetes alapú logaritmusát véve megszabadulhatunk az exponenciális hatványtól:

$$\begin{aligned} -\frac{1}{2} ln\left(|\mathbf{C}_i|\right) + ln\left(P(i)\right) - \frac{1}{2}(\mathbf{x}-\mu_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mu_i) > \\ > -\frac{1}{2} ln\left(|\mathbf{C}_j|\right) + ln\left(P(j)\right) - \frac{1}{2}(\mathbf{x}-\mu_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mu_j). \end{aligned}$$

Mindkét oldalt -2-vel szorozva megfordul a relációs jel:

$$\begin{aligned} ln\left(|\mathbf{C}_i|\right) - 2ln\left(P(i)\right) + (\mathbf{x}-\mu_i)^T \mathbf{C}_i^{-1}(\mathbf{x}-\mu_i) < \\ < ln\left(|\mathbf{C}_j|\right) - 2ln\left(P(j)\right) + (\mathbf{x}-\mu_j)^T \mathbf{C}_j^{-1}(\mathbf{x}-\mu_j). \end{aligned}$$

Homoszkedaszticitás feltételének teljesülése esetén a kovariancia mátrixok megegyeznek:

$$\mathbf{C} = \mathbf{C}_i = \mathbf{C}_j.$$

Ezzel tovább egyszerűsítve a képletet:

$$ln\left(|\mathbf{C}|\right) - 2ln\left(P(i)\right) + (\mathbf{x}-\mu_i)^T\mathbf{C}^{-1}(\mathbf{x}-\mu_i) <$$
$$< ln\left(|\mathbf{C}|\right) - 2ln\left(P(j)\right) + (\mathbf{x}-\mu_j)^T\mathbf{C}^{-1}(\mathbf{x}-\mu_j).$$

Mátrixszorzás szabályainak megfelelően a $(\mathbf{x}-\mu)^T\mathbf{C}^{-1}(\mathbf{x}-\mu)$ tagot felbonthatjuk.

$$(\mathbf{x}-\mu)^T\mathbf{C}^{-1}(\mathbf{x}-\mu) = \mathbf{x}\mathbf{C}^{-1}\mathbf{x}^T - 2\mu\mathbf{C}^{-1}\mathbf{x}^T + \mu\mathbf{C}^{-1}\mu^T$$

Így az egyenlőtlenség:

$$ln\left(|\mathbf{C}|\right) - 2ln\left(P(i)\right) + \mathbf{x}\mathbf{C}^{-1}\mathbf{x}^T - 2\mu_\mathbf{i}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{i}\mathbf{C}^{-1}{\mu_\mathbf{i}}^T <$$
$$< ln\left(|\mathbf{C}|\right) - 2ln\left(P(j)\right) + \mathbf{x}\mathbf{C}^{-1}\mathbf{x}^T - 2\mu_\mathbf{j}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{j}\mathbf{C}^{-1}{\mu_\mathbf{j}}^T.$$

Az összegzéstől független tagokat kivonhatjuk az egyenlőtlenség mindkét oldalából.

$$-2ln\left(P(i)\right) - 2\mu_\mathbf{i}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{i}\mathbf{C}^{-1}{\mu_\mathbf{i}}^T <$$
$$< -2ln\left(P(j)\right) - 2\mu_\mathbf{j}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{j}\mathbf{C}^{-1}{\mu_\mathbf{j}}^T.$$

Leosztva (-2)-vel az relációsjel ismét megfordul:

$$ln\left(P(i)\right) + \mu_\mathbf{i}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{i}\mathbf{C}^{-1}{\mu_\mathbf{i}}^T > ln\left(P(j)\right) + \mu_\mathbf{j}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{j}\mathbf{C}^{-1}{\mu_\mathbf{j}}^T.$$

Természetesen az összes egyenlőtlenség az $i \neq j$ feltétel mellett igaz. Ezzel definiálhatjuk a diszrkimináló függvényünket,

$$f_i := \mu_\mathbf{i}\mathbf{C}^{-1}\mathbf{x}^T + \mu_\mathbf{i}\mathbf{C}^{-1}{\mu_\mathbf{i}}^T + ln\left(P(i)\right).$$

Így minden osztályhoz tartozó diszkrimináló függvényünket kiszámíthatuk minden megfigyelésre. Tehát egy megfigyelés $\mathbf{x}$ mért értékekkel az i osztálycímkét kapja, ha

$$f_i > f_j, \qquad \forall i \neq j.$$

4.3. Döntési fák

Az osztályozási feladatok egyik legismertebb és legszélesebb körben használt eszköze a döntési fa, amely a problémát egyszerű döntések sorozatára vezeti vissza. Ezt egy fa struktúrába illesztve, a gyökértől lefelé haladva és minden csúcsban egy attribútum értéket vizsgálva eljutunk az osztálycímkét megadó levelekig. Így egy új megfigyelést fentről lefelé, minden csúcsban a megfelelő attribútum értékek szerint haladva sorolhatjuk be egy osztályba. A döntési fák számos jó tulajdonsággal rendelkeznek, automatikusan felismernek egyes lényeges változókat és a gyökér közelében helyezik el, a kevésbé lényegeseket pedig a levelek közelében. Előfordulhat, hogy egyes attribútumok nem jelennek meg a fában, mivel nem befolyásolják a döntést. Ennek fényében a döntési fák dimenziócsökkentésre is alkalmasak. Előnyük még a skálázhatóság, vagyis hatékonyan alkalmazhatóak nagy adatminta esetén is. Sokszor használunk bináris fákat a gyakorlatban, ami azt jelenti, hogy minden csúcsnak két gyermeke van. Könnyű belátni, hogy bármely nem bináris fa átalakítható binárissá.

A fa felépítéséhez szükségünk van tanító adathalmazra a megfelelő osztálycímkékkel, így ez a módszer is felügyelt tanulás. Ezt hívjuk partíciós folyamatnak. Válasszuk ki azt az attribútumot, amely a legjobban szeparálja az adathalmazt és ez kerül a fa gyökerébe. Az attribútum minden egyes lehetséges értéke egy elágazást ad és ezek szerint osztjuk szét a mintánkat. Az algoritmus az összes többi csúcsra rekurzívan alkalmazza ugyanezt az eljárást a már csökkentett attribútumhalmazt használva, tehát ha egy attribútumot már felhasználtunk, akkor azt már egyetlen leszármazottjánál sem kell figyelembe venni. (Vannak algoritmusok, amelyek megengedik az egy attribútum szerinti többszöri bontást.) Ha egy csúcsban szereplő megfigyelések mindegyike ugyanazt az osztálycímkét tartalmazza, a csúcsból levél lesz, ezzel az osztálycímkével. Az algoritmus akkor áll le, ha teljesül az alábbi feltételek valamelyike:

- A fa mélysége elért egy előre megadott határt.
- Egy adott csúcs minden eleme ugyanazt az osztálycímkét hordozza.
- Egy előre megadott korlátnál kevesebb megfigyelést tartalmaz a csúcs.
- Nem maradt olyan attribútum, ami szerint tovább lehetne bontani az adathalmazt. Ebben az esetben ez a csúcs levél lesz, az osztálycímke pedig a többségi szavazás elvén az lesz, amiből a legtöbb van az ehhez a ponthoz tartozó adathalmazban.

Hogyan döntjük el, hogy melyik attribútom szeparálja legjobban az adatunkat az adott csúcsban? Ez már algoritmusfüggő, de leggyakrabban a Gini-indexet és az információnyereség elvét használják.

4.3.1. CART algoritmus

Döntési fák előállítására gyakran használt módszer a CART algoritmus (Classification And Regression Trees). A fa építésénél a CART bináris döntéseket használ, és a tesztattribútum kiválasztásához Gini-indexet számol az egyes csomópontokban.

4.1. Definíció. Legyen S egy adathalmaz, amely N mintát tartalmaz. Ezeket a mintákat soroljuk n különböző osztályba. Ekkor S adathalmaz Gini-indexe az alábbi összefüggés alapján számítható:

$$gini(S) = 1 - \sum_{i=1}^{n} p_i^2,$$

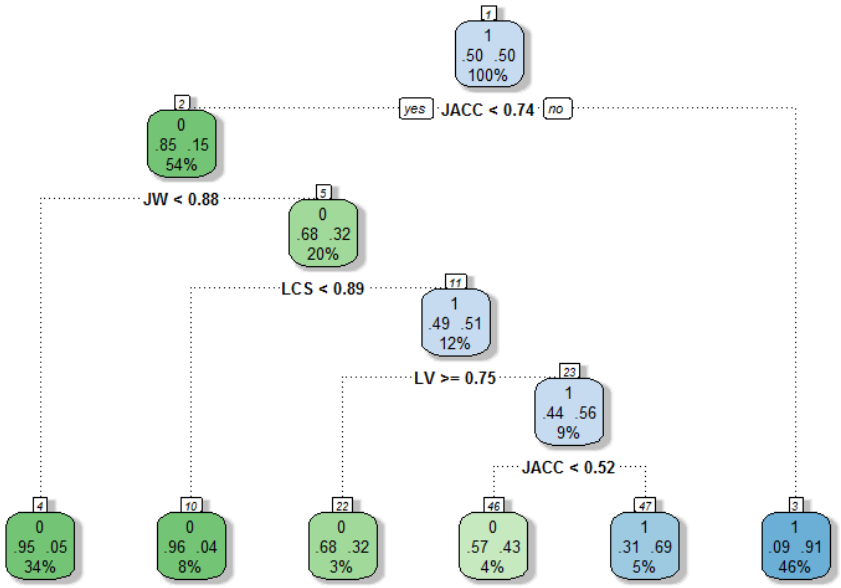
ahol p_i az i. osztály relatív előfordulását jelöli az S halmazban.

Osszuk fel az éppen vizsgált S halmazt egy N_1 elemszámú S_1 és egy N_2 elemszámú S_2 halmazra. Ekkor a Gini-index:

$$gini(S) = \frac{N_1}{N} \cdot gini(S_1) + \frac{N_2}{N} \cdot gini(S_2).$$

Itt $gini(S_1)$ és $gini(S_2)$ értékeket a definícióból számíthatjuk ki.

Az adott csomópontban határozzuk meg minden attribútumra a Gini-index értékét, és válasszuk ki azt, amelyiknél ez az érték a legkisebb.



4. ábra. Példa egy döntési fára a konkrét feladatból

4.4. Random Forest

Az ötlet a módszer megalkotása mögött az volt, hogy több tanuló modell kombinálása növeli az osztályozás hatékonyságát. Random Forrest egy lényeges változtata a bagging módszernek és n darab döntési fa összességét értjük alatta. Az algoritmus bootstrap módszerrel kiválasztja egy Z részhalmazát N tanító adathalmazunknak és épít belőle egy fát, majd ezt megismétli n-szer. Ezt követően a teszteseteket minden generált döntési fán futtatja, és az előrejelzés a fák eredményeinek átlaga lesz, illetve diszkrét osztályozó esetén a módusza. Illetve ez már implementálásfüggő, diszkrét osztályozó esetén is visszatérhet átlaggal a modell. A bootstrap a tanító adatot ismétléses

kiválasztással veszi a rekordokból egyenletes eloszlás szerint. Annak valószínűsége, hogy egy rekord bekerül a tanító adathalmazba:

$$1 - \left(1 - \frac{1}{N}\right)^n,$$

ahol N a rekordok száma, n pedig a választásoké. Ez $n \approx N$ és $N \to \infty$ esetén az ismert határérték szerint $1 - \left(1 - \frac{1}{N}\right)^n \to 1 - \frac{1}{e} \approx 0.632$. Nagy előnye, hogy nem kell számontartani, hogy miket választottunk ki és miket nem.

4.5. AdaBoost

Az Adaboost (Adaptive Boosting) szintén egy együttes osztályozó, tehát több algoritmust használ. Abban különbözik a Random Forest-től, hogy míg az előbbi egy egyneletes eloszlás szerinti bootstrappet követ, úgy az Adaboost a mintavételezést az alábbi iteratív módon teszi:

- Súlyokat rendel minden rekordhoz, ami az első lépésben egyenletesen oszlik el, tehát minden rekord ugyanazt a súlyt kapja.
- Aktuális súlyok szerint végez egy mintavételezést.
- Egyszerűen lefuttatja a 'gyenge' algoritmust, például - de nem feltétlenül - egy döntési fát, majd kiértékeli.
- A helytelenül becsült sorokhoz tartozó súlyokat megnöveli, így a nehezen becsülhető rekordok egyre nagyobb súlyt kapnak.
- Egy előre meghatározott számú iteráció után az algoritmus leáll, és megtörténik az összegzés. A végső becsült érték az így létrejött modellek becsléseinek súlyozott átlaga lesz.

4.6. Teljesítménymutatók

A legfontosabb mutatószám az osztályozó pontossága, ahol összehasonlítjuk a teszt minták valós osztálycímkéit az osztályozó által jósolt osztálycímkékkel, méghozzá úgy, hogy egy mátrixba rögzítjük a kapott eredményeket. A mátrix $n \times n$-es, ahol n az osztályok száma. Az i-edik sor j-edik eleme adja meg azoknak a megfigyeléseknek a számát, amelyeket az osztályozó a j-edik osztályba sorolt, miközben az i-edik osztály címkéjét viselik. A főátlóban találjuk a helyesen osztályozott rekordok számát. Bináris esetben a mátrix a következő:

	1	0
1	TP	FN
0	FP	TN

5. ábra. Általános tévesztésmátrix bináris esetben

A 'True-False', és a 'Positive-Negative' angol szavak variációiból kapjuk a négyféle kategóriát. Értelemszerűen a 'True' kezdetűek jelentik a helyesen klasszifikált pontok számát. A tévesztésmátrix alapján könnyen számítható az osztályozó modellünk pontossága:

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

Ezekből több különböző mutatószám is számítható:

- Ténylegesen negatív arány (Specificity):

$$\frac{TN}{TN + FP}$$

- Precizitás (Precision):

$$\frac{TP}{TP + FP}$$

- Felidézés (Recall):

$$\frac{TP}{TP + FN}$$

- Hamis negatív arány:

$$\frac{FN}{TP + FN}$$

4.6.1. Kappa statisztika

A pontosság megtévesztő lehet, önmagában nem lehet vele jól mérni egy osztályozó teljesítményét. Legyen egy bináris célváltozónk, mely 93%-os valószínűséggel veszi fel az egyik értéket és 7%-ossal a másikat. Ebben az esetben egy 85%-os pontosságú modell is gyenge, hiszen nagyobb pontosság érhető el a véletlen osztályozóval ($\sim$ 87%). Ezt veszi figyelembe a kappa statisztika, ami az osztályozót a véletlen osztályozóhoz hasonlítja.

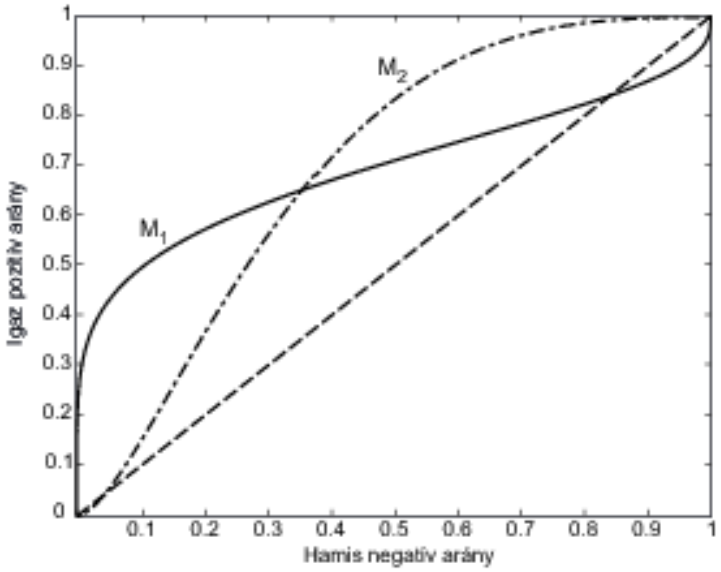
4.2. Definíció. Egy osztályozó kappa statisztikája:

$$\frac{T - \sum_{i=1}^{k} n_i p_i}{\sum_{i=1}^{k} n_i - \sum_{i=1}^{k} n_i p_i},$$

ahol p_i az i. osztály relatív gyakoriságát, n_i az i. osztály előfordulását, T pedig a helyesen klasszifikált pontok számát jelöli. Értékkészlete 0 és 1 közötti, 0 ha véletlen osztályozó és 1 ha tökéletes.

4.6.2. ROC-görbe

A ROC-görbe egy grafikus módszer, amit egy osztályozó igaz pozitív aránya és hamis negatíg aránya közti kompromisszum megjelenítésére használunk. A ROC görbe rajzolásához az osztályozónak képesnek kell lennie egy folytonos kimenetet előállítani, mint például egy LDA osztályozó vagy egy Naive Bayes szerinti posteriori valószínűség, esetleg egy Random Forest diszkrét osztályozó átlaggal visszatért értéke.



6. ábra. Két különböző osztályozó ROC görbéje [6]

A 6. ábrán látható görbék egyes pontjai a megfelelő osztályozó által indukált modellek bizonyos vágás szerinti eredményeit mutatják. Vágás alatt azt értjük, hogy milyen érték felett tekintjük az előrejelzést P-nek, és így alatta értelemszerűen N-nek. Tehát például, ha az LDA posteriori valószínűségeit sorbarendezzük, majd minden egyes értékre kiszámoljuk az osztályozó igaz pozitív arányát és hamis negatív arányát úgy, hogy pont az az érték a vágás, akkor ezen teljesítménymutatók értékei kirajzolják a görbét. Használhatjuk osztályozók összehasonlítására, valamint az ideális vágási pont meghatározására. Láthatjuk hogy M_1 és M_2 nem dominálják egymást, vágástól függően teljesít jobban az egyik, illetve a másik modell. Annál jobb az osztályozó, minél közelebb helyezkedik el a görbéje a bal felső sarokhoz. A szaggatott vonal, tehát a $y = x$ egyenes pedig a véletlen osztályozót reprezentáló egyenes.

4.6.3. AUC

Az AUC (Area Under Curve) a ROC görbe alatti terület. Megadja, hogy egy véletlen pozitív példa milyen valószínűséggel van előrébb, mint egy véletlen negatív. Tökéletes modell esetén 1, a véletlen osztályozó esetében pedig $0,5$. Ez a mérőszám viszont már rendezhetően megkülönbözteti a modellek jóságát, nagyobb AUC jobb modellt jelent.

4.7. Teljesítmény mérésének módszerei

Az osztályozók pontossága természetesen függ attól, hogy milyen adathalmazon mérjük vissza a teljesítményt, vagyis attól, hogy hogyan választjuk szét az ismert osztálycímkéjű adatunkat tanító és tesztelő mintára. Számos módszer ismert az adatok szétválasztására:

- Véletlen vágás: A rendelkezésünkre álló címkézett adatot két diszjunkt csoportra osztjuk valamilyen általunk megadott arányban. Az így kapott két adathalmaz egyikén tanítjuk az algoritmust, a másikon teszteljük. Általában célszerű a nagyobbon tanítani, hiszen kis számú mintánnál fenn áll az alulillesztés veszélye, vagyis nincs elég megfigyelésünk ahhoz, hogy az algoritmus megtanulja az adatunk sajátosságait.

- Keresztvalidáció: Az S adathalmazunkat felbonjuk k darab azonos méretű diszjunkt részhalmazra. Ezekből egyiken tesztelünk, a maradék $k-1$-en pedig tanítunk, ezt követően egy másikon tesztelünk és megint a maradékon tanítunk és ezt folytatjuk addig amíg már az összes részhalmazon teszteltünk. Így tehát k darab modellt építettünk, és a végső teljesítmény meghatározásánál ezen modellek teljesítményének átlagát vesszük.

- Leave-one-out: Ez a keresztvalidáció speciális esete, ahol $|S| = k$ tehát

minden részhalmazunk egy rekordból fog állni. Előnye, hogy nem szerepel benne a véletlen faktor, viszont hátránya a hatékonyság, hiszen nagy adathalmaz esetén nagyon sokszor kéne modellt építeni.

5. A feladat megoldása

A szükséges elméleti ismeretek bemutatása után térjünk át a konkrét feladatra. Egy feloldással rendelkező adathalmazon szeretnénk tanítani egy modellt, ami minél jobb teljesítménnyel képes megtalálni két különböző forrásból érkező adattáblákban azokat a cégeket, amik mindkettőben előfordulnak. A modellezés lépései:

1. Adatfeltárás és előkészítés, tisztított cégnevek előállítása
2. Sztringtávolságok kiszámítása a rendelkezésünkre álló címkézett adathalmazunk megadott párjaira
3. Változószelekció
4. Ismert osztálycímkéjű adatunk kettéválasztása, tanító és tesztelő adathalmazra
5. Modellek illesztése
6. Modellek kiértékelése
7. Modell választása
8. A két adatbázisból minden lehetséges pár megalkotása
9. Ezen párokra a sztringtávolságok kiszámítása
10. Kiválasztott modell alkalmazása az adatra

5.1. Adatfeltárás és előkészítés

Alapvetően négy adatforrásból fogunk dolgozni, a két adathalmazból alkotott párokra fogjuk futtatni az osztályozónkat, míg rendelkezésünkre áll egy múltbéli ellenőrzött adattábla, ahol két gyanúsan hasonló cégnév mellett

megtalálható a feloldás is, tehát hogy valóban egy entitásra utalnak-e. Ezt fogjuk használni tanító adatnak, amin az algoritmusunk megismeri az adatok sajátosságait. A tanító adatunk 1231 rekordból áll, amiből 621 hordoz IGEN és 610 NEM feloldást. A két adathalmaz egyenként 10000 rekordból áll. Valamint használok egy kisebb, egyoszlopos adattáblát, ami a gyakran előforduló, plusz információval nem rendelkező szavakat tartalmazza, amiket törölhetünk a cégek nevéből. Mind az adattisztítást, mind a modellezést az R programmal végeztem [10].

	1. név	2. név	Feloldás
1.	Arsenal Ltd.	Arsenal LIMITED	YES
2.	Jeremy Ahern Vet	Jeremy Aherne Vet	YES
3.	Little Milano Take Away	L. Milano Takeaway	YES
4.	Relish Cafe	Delish Cafe	NO
5.	McCleans	Thomas McClean And Sons	NO

7. ábra. Cégnevek és feloldásuk, részlet a tanító adatunkból

Mint minden modellező feladatnak, úgy ennek is az első lépése az adatok megismerése és előkészítése. Mivel sztringekkel dolgozunk, érdemes vetni egy pillantást az adat minőségére karakterek és szavak szintjén is.

5.1.1. Karakter szintű tisztítás

Az adatról már első pillanatban látszik, hogy tartalmaznak speciális karaktereket, amik az automatizált rendszereknek köszönhetően generálódhattak. Ezek csökkenthetik a sztringhasonlóságok értékét, így szeretnénk tőlük megszabadulni. A kis-nagy betű különbségeket is szeretnénk eltüntetni, ezért az összes betűt nagybetűvé konvertáljuk, valamint töröljük a whitespaceket is. Az R programozási nyelv számos módszerrel rendelkezik ezek elérésére. Én a

grepl() függvényt használtam 'regex' kifejezésekkel. Ezeket a módszereket mindkét táblára és a tanító adatunkra alkalmazva egy lépéssel közelebb kerülünk a megoldáshoz. Mivel az n-gram alapú sztringtávolságokat nem lehet alkalmazni $n-1$ vagy annál rövidebb karakterhosszúságú cégnevekre, ezért azokat kiterjesztem, úgy, hogy n hosszúak legyenek, vagyis r hosszú cégnév mögé $n-r$ darab '.' karaktert illesztek. Mivel ezeket már kiszűrtük, ezzel a módszerrel nem növelem a sztringhasonlóságok mértékét manuálisan, ami félreosztályzáshoz vezethetne.

5.1.2. Szó szintű tisztítás

Észrevehetjük, hogy némely cégnév mögött szerepel a cégforma, máshol nem, vagy esetleg rövidítve. Talán ez is okozhatja, hogy egy cég neve különbözően jelenik meg a két adatbázisban? Ezen gondolatmenetet tovább boncolgatva rájöhetünk, hogy érdemes lenne megtalálni azokat a szavakat, amiket eltávolítva nőhet majd a modellünk hatékonysága. Ezt többféleképpen meglehet tenni, én gyakoriság alapján szűrtem ki a szavakat. Ha az összes rekord több, mint x%-ánál szerepelt egy szó, akkor törlésre került. Így x a modellünk paraméterévé vált. Ebben az R 'tm' nevű csomagja volt segítségemre. Az így kigyűjtött szavak kerültek bele a negyedik adattáblánkba. Azért tartottam szükségesnek külön kimenteni ezt listát, hogy esetleg manuális bővítésre is legyen lehetőség az adatok alaposabb ismerete után. Majd ezen listában található összes szót töröltük, mind az osztálycímkével rendelkező, mind az azt nélkülöző adatainkból.

	Cégnév	Tiszta Cégnév
1.	Arsenal Ltd.	ARSENAL
2.	Mace & Post Office	MACEPOST

8. ábra. Tisztított cégnevek, részlet az adatunkból

5.2. Távolságok számítása

A fentebb bemutatott sztringtávolságokból értelemszerűen nem használtuk mindegyiket, mert némelyik csak egy apróbb változtatást tartalmaz egy másikhoz képest, vagy felvetődött hogy nem alkalmasak az adott feladatra. Valamint szempont volt a hatékonyság és az egyszerű implementálhatóság is. A hatékonyságot kiemelten fontosnak tartottam, hiszen a modell alkalmazása előtti adatelőkészítés futási ideje $\mathcal{O}(m \cdot n \cdot i \cdot k)$, ahol m és n a két adatbázis elemszáma, i a számolt sztringmetrika száma, míg k a sztringmetrikák kiszámításának átlagos ideje. Az implementációt az R 'stringdist' nevű csomagjának segítségével végeztem, méghozzá úgy, hogy minden távolságot hasonlósággá konvertáltam, és $[0, 1]$ közé normáltam. Tehát, ha bármelyik sztringmetrika az 1 értéket vette fel, akkor az teljes egyezést jelentett, a 0 pedig teljes különbözőséget. A leghosszabb közös szövegrészhez beépített csomagot nem találtam, így leprogramoztam magam. A pszeudokód látható a 3. fejezetben, a távolság leírása alatt.

	1. Cégnév	2. Cégnév	LV	OSA	DL	JAC
1.	Donati Holding	Donati Ltd.	0,46	0,46	0,46	0,42

	1. Cégnév	2. Cégnév	LCS	JAR	JW
1.	Donati Holding	Donati Ltd.	0,63	0,82	0,89

9. ábra. Példa egy cégnévpáron kiszámított hasonlóságokra

5.3. Változószelekció

A sztringtávolságok lényegében ugyanazt mérik, két sztring hasonlóságát, csak éppen különböző módon és különböző szempontok szerint. Így két, közel megegyező sztringre mindegyik magas, két szinte teljesen különbözőre mindegyik alacsony hasonlóságot fog adni. Ez azt is jelenti, hogy a lineáris korreláció a magyarázó változóink között magas. Például a LV és DL értékek lineáris korrelációja a tanító adatunkban: $0,998$. Ez várható is volt, hiszen a két metrika számítása alig tér el egymástól. Ezért célszerű olyan változókat választani, amelyek a leginkább függetlenek egymástól a számítási módjukat illetően. A CART módszer egyik fontos tulajdonsága, hogy a klasszifikációs és regressziós fák szerkezete invariáns a független változók monoton transzformációjára. Automatikusan felismeri az interakciókat és a nonlinearitást is jól kezeli. Ezért az irreleváns változók nem kerülnek bele a fába. Így a változószelekciót a fára bíztam. A hipotézisünk beigazolódott, a fába került változók: LV, JAC, JW, LCS. A Levenshtein egy szerkesztési műveletek alapú, a Jaccard egy n-gram alapú, míg bár a Jaro-Winkler is szerkesztési művelet alapú, a súlyozás a sztringek elején levő hibák tekintetében elegendő eltérést biztosít. Az Leghoszabb Közös Szövegrész (LCS) pedig a normálás után visszaadja, hogy a rövidebb sztring hány százaléka található meg a hosszabb sztringben. Így a kezdeti hipotézisemet a döntési fa igazolta.

5.4. Modellek illesztése, kiértékelés és teljesítmény

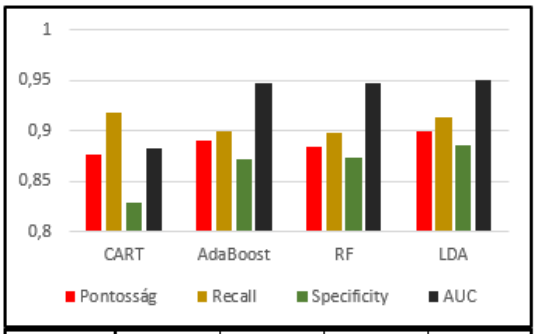
Minden modellt a R 'caret' nevű csomag train() függvényével építettem, a megfelelő paraméterekkel. Minden módszer esetében az adatokat véletlen módon kettévágtam $2:1$ arányban és az adatok $\frac{2}{3}$-án tanítottam, míg az $\frac{1}{3}$-án validáltam, majd ezt a folyamatot 15-ször megismételtem hogy a véletlen faktort csökkentsem. A teljesítmény mutatók az így felépült 15 modell teljesítményének átlagai minden módszer esetében. Az első modellek tévesztés-

mátrixai a 10. táblázatban láthatóak. Vízszintesen a tényleges értékek, míg függőlegesen a módszerek által prediktáltak láthatók.

	CART		AdaBoost		RF		LDA	
	I	**N**	**I**	**N**	**I**	**N**	**I**	**N**
I	176	34	183	27	183	27	186	24
N	15	192	21	186	21	186	18	189

10. ábra. Első vágás utáni tévesztésmátrixok

Már ebből is tisztán látszik, hogy az LDA modellünk várhatóan jobban fog teljesíteni, mint a többi, döntési fa alapú modellek. A fontosabb teljesítmény mutatók pedig egyértelműsítik ezt, miszerint a Lineáris Diszkriminancia Analízis erősebb modellt hozott létre. Meg kell említeni, hogy a többi módszer is kiváló eredménnyel zárt.



	CART	AdaBoost	RF	LDA
Pontosság	0,8765	0,8908	0,8844	0,8994
Recall	0,9175	0,8991	0,8981	0,9129
Specificity	0,8281	0,8716	0,8736	0,8855
AUC	0,8828	0,9467	0,9467	0,9509

11. ábra. Módszerek átlagos teljesítménymutatói

5.5. Adatunk felcímkézése

Kész a modellünk, nincs más hátra mint eldönteni, hogy melyik külön adatbázisban szereplő párokról mondhatjuk, hogy ugyanazon cégre utalnak. Ezt úgy tettük meg, hogy elkészítettük az összes lehetséges párt, és kiszámoltuk rájuk a kiválasztott sztringhasonlóságokat és az adatot ráfuttattuk a modellünkre, ami beosztályozta őket megegyezőnek (IGEN), illetve különbözőnek (NEM). Az elkészíthető 100 millió párból 147 párt talált megegyezőnek az algoritmusunk. Mivel létező cégnevekből összeállított, 100%-os egyezésű rekordokat nem tartalmazó adattáblákkal dolgoztam, ez a szám tükrözi az elvárásokat. A nagyságrend természetesen az adat sajátosságától függ.

Megemlítenék itt egy gyakorlati problémát, amivel sokszor kell szembesülnünk, ha nagy méretű adattáblákat szeretnénk összehasonlítani. A futási idő a táblák növelésével négyzetesen változik. Gondoljunk csak bele, ha csak 1 darab új nevet tennénk az egyik táblába, a keletkező új párok száma a másik tábla elemszáma lenne. Ezért több tízezres nagyságrendű tábláknál a futási idő, implementálástól és a számítógép paramétereitől függően akár több napig is eltarthat. Ennek a problémának a megoldásával a dolgozat nem foglalkozik.

6. Összegzés

A feladat során használt matematikai tartalom megismerése után láttuk, hogy a sztring alapú hasonlóságok és az adatbányászat módszereit ötvözve sikeres megoldást találtunk egy valós, sok helyen előforduló problémára. A szöveges változóink leképezése sztringtávolságokká - tehát numerikus változókká - lehetővé tette gépi tanulási módszerek alkalmazását. A modellünk képes hatékonyan leutánozni az emberi agy felismerőképességét, ennek fényében pedig elmondhatjuk, hogy automatizáltuk a feladat megoldását. Bár nem képes olyan teljesítményre mint az ember, és az eredmények felülvizsgálást igényelnek, de ez az automatizált megoldás rengeteg időt és pénzt spórolhat a felhasználónak. A megoldás váza könnyedén átültethető más, hasonló jellegű problémák esetére is.

Hivatkozások

[1] Lise Getoor, Ashwin Machanavajjhala. *Entity Resolution: Tutorial*. 2012. https://www.umiacs.umd.edu/~getoor/Tutorials/ER_VLDB2012.pdf

[2] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis and Vassilios S. Verykios. *Duplicate Record Detection: A Survey*. IEEE Transactions On Knowledge And Data Engineering, Vol. 19, No. 1, January 2007. https://archive.nyu.edu/bitstream/2451/14760/5/tkde2007.pdf

[3] Damerau, Fred J. *A technique for computer detection and correction of spelling errors*. March 1964.

[4] W. B. Canvar and J. M. Trenkle. *N-gram-based text categorization*. In Proc. of SDAIR-94, 3rd Annual Symp. on Document Analysis and Information Retrieval, pp. 161–175, Las Vegas, USA, 1994.

[5] Dr Sami Khuri, *Longest Common Substring*. CS:255 Design and Analysis of Algorithms Computer Science Department, San Jose State University. https://avinashanantharamu.files.wordpress.com/2013/02/longest-common-substring.pdf

[6] Tan, Pang-Ning; Steinbach, Michael; Kumar, Vipin *Introduction to Data Mining*. 2005.

[7] Han, Jiawei, Jian Pei and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[8] Teknomo, Kardi. *Discriminant Analysis Tutorial*. 2015.

[9] Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The Elements of Statistical Learning*. Stanford, California August 2008.

[10] The R Foundation, *R version 3.3.3 (Another Canoe) for Windows*. https://www.r-project.org/