\*quick draft\*

Ez olyan mértékű draft, hogy ezt biztos át kell írnod **Artúr.**

Kérdésfelvetés: Mindig is kulcsfontosságú volt az egyetemeken a lemorzsolódás csökkentése, hisz ez is egy jó képzés egyik ismérve. Az egyetem egy teljesen új környezet a középiskolaihoz képest, és sokaknak nehéz ilyen gyorsan mentalitást váltani, ami miatt az első évben nehézségeik akadhatnak. Ezért minél korábban, minél több információ alapján szeretnénk behatárolni, hogy melyek azok a hallgatók, akik esetleg segítségre szorulhatnak egyetemi tanulmányaik elkezdése során.

Adattudományi szempontból tehát azt kell vizsgálni, hogy az egyetemi karriert megkezdődően a felvételi pontszám, 0. ZH pontszám és kognitív teszt alapján jó predikciót tudjunk adni arra, hogy adott tárgyakból milyen eredményt, jegyet fognak elérni az egyes hallgatók, tehát ez egy többosztályos osztályozási feladat.

Adathalmaz jellemzése: egyrészt évekre lebontva megkaptuk a kognitív teszt eredményeit, 0.ZH pontszámokat, felvételi pontszámokat és tantárgyi eredményeket, mindegyiket külön-külön táblában. A táblákat az alapján tudjuk illeszteni, hogy a hallgatók neve és Neptun kódja szerepel. Érdemes hangsúlyozni, hogy a hallgatók közül nem mind írt kognitív tesztet, így, ha azt is be akarjuk venni a modellezésbe a többi attribútum mellé, akkor kevesebb adatpontunk lesz.

**Adathalmazok jellemzése, előfeldolgozási lépések**

Az adatokat Szilágyi Brigitta Tanárnő bocsátotta a rendelkezésünkre. A 2019, 2020 és 2021-ben kezdett hallgatókról is voltak adataink, azonban mi a vizsgálódásaink nagy részét a 2019-es adatokra szorítkozva végeztük el, mert csak azok voltak kellően részletesek (a feladat kezdeti fázisában azonban a többi évhez kötődő adattáblán is hajtottunk végre adattisztítási lépéseket, csak később ezt nem tudtuk hasznosítani). 2019-hez kapcsolódóan több adattáblánk volt, az egyikben a hallgatók felvételi eredménye szerepelt, a másikban a 0.ZH pontszámuk, egy harmadikban a kognitív teszten elért eredményeik, illetve egy negyedikben a Matematika A1a és A2c osztályzataik, illetve még néhány, ahonnan 1-1 adatot kellett kinyernünk. Ahhoz, hogy ezeken dolgozni tudjunk, nyilvánvalóan össze kellett fűznünk őket a hallgatók neve és Neptun kódja alapján. Azonban ez előtt célszerű volt az egyes táblákat egyenként megvizsgálni és alakítani, hogy minél több irreleváns oszlopot ki tudjunk szűrni, így csökkentve a keletkező összefűzött adattábla méretét.

A kognitív eredményeket tartalmazó adattábla részletesen tartalmazott információt egyrészt minden hallgatóról (hova valósi, emelt érettségit tett-e matematikából, reál tagozatos volt-e, milyen szakra és tankörbe jár), másrészt a hallgató teszteredményéről is (mennyi idő alatt töltötte ki a tesztet, mely kérdéseket válaszolta meg jól, milyen lett a nyelvi és matekos teljesítménye), illetve tartalmazott néhány, a teszthez kapcsolódó egyéb információt is (például a teszthez használt edubase jelszó, felhasználónév). Természetesen nekünk ennyi adat nem kell, úgyhogy ebből az adattáblából jó pár irreleváns oszlopot ki kellett szűrnünk. Amelyeket meghagytunk, azok az alábbiak: a hallgató neve (ez már elég volt, hogy csak ez alapján fűzzük össze a táblákat) és Neptun kódja; emelt érettségit tett-e matematikából; reál/matematika tagozatos volt-e; szak és tankör; az elért pont és százalékos teljesítmény a nyelvi és matekos részben, valamint összességében. Problémát jelentett még, hogy a ’Szak’ mezőben mindenki másképp írta be azt, hogy melyik szakon tanul, így ezt szabványosítani kellett, ha később szakok szerint akartunk vizsgálódni. Erre a feladatra külön Python kódot írtunk, és ha volt olyan mező, ahol nem tudtuk eldönteni, hogy mi lenne az oda tartozó érték (például mert ’VBK’ volt odaírva), arra bevezettünk egy globális ’UNKNOWN’ változóértéket, azonban szerencsére ilyenből kevés volt. Kicsit még tisztítani kellett a ’Tankör’ értékeken is, de mivel ilyenből kevés volt, ezt manuális is meg tudtuk tenni.

A 0. ZH eredményeket tartalmazó tábla szerencsére ennél jóval kisebb volt, csak a hallgató nevét, Neptun kódját, képzés nevét, illetve kódját, felvétel évét, valamint a ZH eredményt tartalmazta. Ebből értelemszerűen csak a névre és az eredményre volt szükségünk, a többi elhagyható.

A felvételi pontszámokat bontva (hozott pont, érettségi, többletpont) tartalmazó tábla hallgatók nevén és születési dátumán kívül tartalmazott még pár, a felvételi eljáráshoz és felvételi döntéshez kapcsolódó adatot, illetve a ponthatárt. Ebből a táblából csak a név és a pontokat tartalmazó oszlopok kellettek, a többit elvethettük.

A Matematika A1a és A2c jegyeket tartalmazó táblával már több dolgunk volt, mint az előző kettő esetben. Először is minden személyhez több rekord tartozhatott, legalább egy az A1a jegyhez, lehetett még egy az A2c jegyhez (nyilván csak azoknak, akik elvégezték az A1a-t, és ott maradtak az egyetemen), illetve, ha egy korábbi, nem a végleges jegyet eredményező vizsgán egy hallgató megbukott, akkor ahhoz is tartozott egy rekord. Az attribútumok között a hallgató nevén, Neptun kódján és az osztályzatán kívül szerepelt még a felvétel éve, képzés neve, kódja, státusz ID (Aktív, elbocsátott stb.), Pü státusz, a tantárgy neve, kódja, kreditértéke, jegy típusa, bejegyzés dátuma, illetve, hogy elismert és hogy érvényes-e az adott jegy. Ezekből az adatokból nekünk csak a hallgató nevére és jegyértékeire volt szükségünk, ráadásul olyan formában, hogy minden sor egy hallgatóhoz tartozzon, és az oszlopok a tantárgyakból szerzett jegyeket tartalmazzák. Ehhez először Pythonban kiszűrtük az irreleváns oszlopokat, majd ’crosstab’-eléssel a kívánt formára hoztuk az adatokat, ahol még ügyelni kellett arra, hogy a korábbi vizsgajegyek ne kerüljenek bele, tehát minden hallgatóhoz tárgyanként csak egy jegy tartozzon. Ezen kívül még, mivel az érdemjegyek szövegesen voltak megadva’, azokat számszerűvé alakítottuk, hogy majd a későbbiekben könnyebb legyen velük dolgozni.

Egy külön adattábla tartalmazta még a kognitív eredményeknél a matekos eredményt blokkokra lebontva, amely valójában az elsőként tekintett adattáblának volt egy egyszerűsített, kevesebb attribútummal bíró változata. Ebből az adattáblából csak a hallgatók nevére, illetve a blokkonkénti teljesítményre volt szükségünk, a többit elhagytuk.

Így már rendelkezésünkre állt az összes tábla, egyenként tisztítva, és már csak az összefűzés volt hátra, amit R-ben könnyen meg tudtunk tenni, valamint még a végén rendeztük az oszlopok sorrendjét, hogy az adathalmaz logikus szerkezetű legyen. Fontos megjegyezni azonban, hogy nem minden elsőéves írt abban az évben kognitív tesztet, így összefűzés során (ahol valójában ’inner-join’-oltunk) kevesebb sorunk lett, mint ahányan abban az évben a BME VBK karára felvételt nyertek.

A legvégén az így kapott adathalmaz 231 rekorddal és 21 oszloppal rendelkezett, amelyek között még esetlegesen szűrtünk különböző algoritmusok használata során.

**Modellezés**

Miután megtörténtek a kellő adatfeldolgozási lépések, nekiláthattunk a modellezésnek is. Tekintve, hogy ’multiclass’ osztályozási feladatról van szó, első megközelítésnek döntési fával próbáltunk dolgozni. **\*Artúr szöveg ide\***

A többi algoritmus használatához még egy kicsit módosítanunk kellett a fő adathalmazunkat. Először kiszűrtük a ’Szak’ attribútum kivételével az összes szöveges értékű attribútumot, valamint eltávolítottuk a redundáns oszlopokat is (pl. az ’Összpont’ és ’Össz teljesítmény’ oszlopokat, hisz ezek a többi oszlopbeli értékből számolhatóak). Ezután két modelltípust hoztunk létre: az egyikben együtt vizsgáltunk minden hallgatót, a másikban különválasztottuk a vegyészmérnököket és a biomérnököket. Sajnos környezetmérnök hallgatóból nem volt annyi, hogy elég jól használható, általános modellt tudjunk rájuk építeni. Továbbá az együttes modellben, mivel nagyon kevés 1-es osztályzat született, SMOTE túlmintavételezési eljárással megpróbáltuk dúsítani az adathalmazt. A szétbontott modellben ezt nem tudtuk megtenni, mert ott csak 1 vagy 2 ilyen címkéjű rekord volt, így azt nem tudtuk volna lekezelni SMOTE-val. Emiatt számolnunk kell azzal, hogy a szétbontott adatmodellen használni kívánt algoritmusok emiatt kissé nehézkesen fognak működni.

Következő lépésként kNN algoritmust használtunk. Mivel szerepeltek erősen korrelált adatok is az adathalmazban, így az euklideszi távolság mellett Mahalanobis távolsággal is próbáltunk modellt építeni. A megfelelő szomszédszám meghatározásához 10-szeres keresztvalidációt alkalmaztunk mindkét esetben, ahol a szomszédszámot 1-től 30-ig iterálva néztük.

Ezután bináris döntési algoritmusokat próbáltunk ki *One VS Rest* alapon. Mindkét adatmodellen azonos algoritmusokat vizsgáltunk, majd összevetettük őket, hogy leellenőrizzük, nyertünk-e azzal, hogy szétbontottuk az adathalmazt.

* Először SVM algoritmust alkalmaztunk. Révén, hogy az adathalmazban kevés diák kapott elégtelen osztályzatot, paraméternek beállítottuk, hogy kiegyenlítetlen osztályozási feladatról lesz szó. Ezután meg kellett határozni a regularizációs tag értékét. 1-től 5-ig 0.1-es beosztással választottuk ki iterálva a megfelelő értéket, és megnéztük 10-szeres keresztvalidációval, hogy az így kapott modell hogy teljesít (5 felett már nem volt nagy változás pontosság tekintetében). Ahol a legmagasabb volt az átlagos pontosság, azt választottuk.
* Ezt követően logisztikus regresszióval próbálkoztunk. Itt is rögzítettük, hogy kiegyensúlyozatlan osztályproblémával állunk szemben, majd megpróbáltuk helyesen beállítani a paramétereket (hibatag, számítási függvény), azonban azt tapasztaltuk, hogy az alapértelmezett értékekkel lesz a legjobb a modell.

Majd kipróbáltuk a lineáris regressziót is, ami bár folytonos célváltozó prediktálására alkalmas a leginkább, úgy alakítottuk át a végeredményt, hogy a becsült értéket kerekítettük, majd megnéztük, ez hogyan illeszkedik a tényleges teszthalmazbeli értékekre. Az átalakítás előtt még megnéztük az és *RMSE* értékeket.

Végezetül még az együttes, nem szétbontott modellre egy Naive Bayes algoritmust is futtattunk (.

A végső, letisztított adathalmaz, amin dolgoztunk, alapból 12 attribútumos rekordokból áll, így azt is megnéztük, hogy főkomponens analízist használva kevesebb, származtatott attribútummal hogyan változik a teljesítmény. Először néhány, intuíciónak megfelelő számú főkomponenssel vizsgáltuk, majd végül végigiteráltunk a lehetséges főkomponens értékeken és megnéztük, hogy adott számú főkomponenssel az eddig használt algoritmusok egy része hogyan teljesít, és ezeket egy táblázatba gyűjtöttük.

Ahogy majd a kiértékelésnél látni fogjuk, ezek változó pontosságú, de nem túl jó végeredményt adtak nekünk. Intuitívan is érződik, hogy csak egyetem előtti és kora egyetemi hallgatói adatokból nagyon pontos végeredményt nem várhatunk, hiszen az első félév során minden hallgató másképp tud megbirkózni az egyetem által rá nehezedő nehézségekkel, és ez a teljesítményeken is tetten érhető: Valaki nagyon könnyen belerázódik az új, egyetemi környezetbe, így képes ugyanolyan jól teljesíteni, mint középiskolában, valaki nehezebben abszolválja, és emiatt gyengébben teljesít, illetve akadhat olyan is, aki a választott szakon sokkal jobban tud teljesíteni, mint ahogy eddig tette. Azonban helytelen lenne azt feltételezni, hogy semmilyen mértékű következtetést nem tudunk levonni ezekből az adatokból, hiszen általában drasztikus változások nincsenek a hallgatók teljesítményében. Így következő lépésnek így módosítottuk a célváltozóértékeket: eddig öt lehetséges értékünk volt, az ötféle osztályzatnak megfelelően. Ezekből *3 új osztályt* hoztunk létre, egyet a 4-es 5-ös hallgatóknak (ők a jó tanulók), egyet a 2-es, 3-as tanulóknak (ők a gyengén teljesítő tanulók), és az utolsót az 1-es érdemjegyű hallgatóknak (akik nem tudták teljesíteni a tárgyat). Így megváltoztatva a célváltozóértékeket, az előzőekhez hasonlóan, kétféle modelltípuson dolgoztunk, szakonként szétválasztva és egyben. Majd ezekre is kipróbáltuk a korábban használt algoritmusokat, és megnéztük, hogy a pontosság mennyivel változott.

**Kiértékelés**

Összességében négy adatmodell-en vizsgáltuk a kNN, lineáris regressziós, Naive Bayes illetve *One VS Rest* elvű bináris osztályozó algoritmusaink teljesítőképességét attól függően, hogy volt-e szakonkénti bontás, illetve, hogy hány lehetséges értéke volt a célváltozónak. Nem meglepő módon, típustól függően meglehetősen eltérő eredményeket kaptunk.

Együttes, 5 célváltozóértékes:   
-   
- Főkomponens analízist használva egyedül az SVM algoritmusnál és a lineáris regressziónál értünk el jobb eredményt, ahol is SVM-nél 3, lineáris regressziónál 4 főkomponenst használva a pontosság 0.35 illetve 0.30 lett, azonban ez nem nevezhető szignifikánsnak.

Szakonként, 5 célváltozóértékes:  
- Itt jobb eredményeket kaptunk, mint az egyesített adatmodellben. Vegyészmérnököknél az alábbi eredmények keletkeztek: Az SVM algoritmus pontossága 0.56 lett (C = 2.5 hibataggal), a logisztikus regresszióé 0.44, a módosított lineáris regresszióé pedig 0.22 ( = 0.36). Biomérnököknél pedig az alábbiak lettek: Az SVM algoritmus 0.32 pontosságú (C = 1.3 hibataggal), a logisztikus regresszióé 0.12, a módosított lineáris regresszióé pedig 0.32 ( = 0.32).  
- Főkomponens analízissel vegyészmérnököknél a lineáris regressziónál tudtunk javítani, főkomponensszámtól függően a pontosság 0.28-0.32-re is felment, illetve a logisztikus regressziónál is kis javítás elérhető. Biomérnököknél viszont logisztikus regresszió pontossága nagy mértékben növelhető, ha 3-5 főkomponenssel dolgozunk, ekkor 0.4 pontosság is elérhető.

Együttes, 3 célváltozóértékes:  
- Ahogy azt sejtettük, kevesebb célváltozóérték esetén javultak a pontosságok. C = 0.7-es hibataggal az SVM algoritmus 0.67-es pontosságot ér el, a logisztikus regresszió 0.61-eset, a lineáris regresszió szintén 0.61-eset. ( = 0.44), a Naive Bayes pedig 0.52-eset.  
- Főkomponensek használatával nagy javulást egyik algoritmusnál sem tudunk elérni, csak a regressziósaknál egy keveset (körülbelül 3 századnyit) de az nem szignifikáns.

Szakonként, 3 célváltozóértékes:  
- A legjobb eredményeket ezzel az adatmodellel tudtuk elérni. Vegyészmérnököknél az alábbi eredményeket kaptuk: SVM algoritmus C = 4.9 hibataggal 0.83-as pontosságot ért el, logisztikus regresszió 0.75-t, és a lineáris regresszió szintén 0.75-öt ( = 0.3186). Biomérnököknél pedig ezeket kaptuk: SVM algoritmus C = 1.3 hibataggal 0.68-as pontosságot ért el, logisztikus regresszió 0.76-ot, és a lineáris regresszió pedig 0.72-t ( = 0.46).  
-Főkomponensek használatával vegyészmérnököknél a regressziós algoritmusoknál akár 0.86-os pontosság is elérhető (3-4 főkomponenssel), biomérnököknél pedig ugyanezn algoritmusoknál a 0.8-as pontosság érhető el (ugyanakkor az értékek mindkét esetben csökkennek).