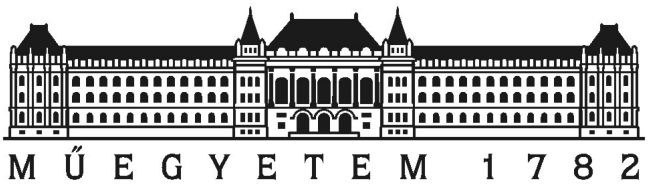


# TDK DOLGOZAT

## Elsőéves VBK hallgatók teljesítményének vizsgálata a Covid előtti és utáni időszakból

**Köller Donát Ákos & Vlaszov Artúr**
BME Matematikus MSc
Adattudomány szakirány

**Témavezető: Szilágyi Brigitta**
Geometria Tanszék


BME Matematika Intézet
Budapest
2022

# Tartalomjegyzék



# 1. Bevezetés

# 2. A kognitív tesztről

# 3. Adatprepació

## 3.1. Az adatok beszerzése, adattáblák jellemzése

## 3.2. Adattisztítás

A felderítő és modellezési fázisban használt adattáblát a rendelkezésre álló táblák megfelelő mértékű tisztításából és összeillesztéséből nyertük.

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 manuálisan 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 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.), Pénzügyi státusz (állami/önköltséges), 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 egysze-

rűsített, 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.

# 4. Felderítő adatelemzés a két évben

# 5. Modellépítés az elsőéves teljesítményekre

## 5.1. Osztályozó algoritmusok

### 5.1.1. kNN

A kNN (k-Nearest Neighbour) algoritmus az egyik legegyszerűbb gépi tanulási algoritmusok közé tartozik. Lényege, hogy az adatponthoz legközelebbi k szomszéd címkeértékének valamilyen távolságalapú súlyozása szerint választjuk meg a kérdéses adatpont címkeértékét.

### 5.1.2. Lineáris regresszió

A lineáris regresszió jellegéből adódóan alapvetően folytonos célváltozó prediktálására alkalmas, ugyanakkor kategorikus, ordinális címkéjű adatok osztályozására is fel lehet használni. Az algoritmus lényege, hogy a magyarázóváltozók mindegyikéhez egy-egy súlyt rendelünk, majd az adatpont attribútumértékeinek vesszük a súlyokkal való súlyozott összegét, és esetleg egy bias tagot hozzáadva az így kapott szumma lesz a prediktált címkeérték. A cél a tanítás során a súlyok optimális megtalálása, miközben a tanítóhalmazon minimalizáljuk a predikciós hibát.

### 5.1.3. Naive Bayes

A Naive Bayes algoritmus működése mögött álló alapelv az, hogy feltesszük az attribútumok feltételes függetlenségét, amennyiben a célváltozó értéke ismert. Osztályozás során azt vizsgáljuk, hogy mely címkeérték mellett a legnagyobb a valószínűsége annak, hogy az adott adatrekord attribútumai éppen a felvett értékeket kapták. A megfelelő valószínűsé-

geket a tanítóhalmazbeli adatok attribútumértékeinek különböző címkék melletti relatív gyakoriságai adják.

### 5.1.4. Gradient Tree Boosting

A Gradient Boosting algoritmus egy Ensemble típusú osztályozó, lényege, hogy sok gyenge teljesítményű prediktor (*weak learner*) eredményét felhasználva

### 5.1.5. Logisztikus regresszió

A logisztikus regresszió alapvetően bináris osztályozási problémák megoldására alkalmas, de kiterjeszthető másfajta feladatok megoldására is. Lényege, hogy a lineáris regresszióhoz hasonlóan az adatrekord attribútumértékeinek súlyozott összegét használjuk egy szigmoid[1] függvény bemeneteként, amely azt utána leképzi az $(0, 1)$ intervallumra, és amennyiben az output 0.5-nél nagyobb, úgy a pozitív osztályba soroljuk az adott rekordot.

### 5.1.6. SVM

Az SVM (Support-Vector-Machine) egy lineáris szeparálást használó bináris osztályozó algoritmus, amely egyéb problémákra is kiterjeszthető. Lényege, hogy különböző magfüggvénye segítségével az adatrekordokat egy magasabb dimenziós térbe képzi le, ahol olyan szeparáló hipersíkot keres, amely maximalizálja a vele párhuzamos, adatpontot nem tartalmazó térrészt. A cél a megfelelő magfüggvény és a szeparáló hipersík megtalálása, amellyel a különböző címkéjű adatpontok lineárisan szeparálhatóak.

## 5.2. Modellek és metodológia

A kutatás során kétféle eredmény alaposabb vizsgálatára összpontosítottunk mindkét évben: a "Matematika A1a - Analízis" tárgyból kapott érdemjegyre illetve az első félév végén megállapított kumulált tanulmányi átlagra. A kettőből az előbbi különös fontossággal bír, ugyanis ebből a tantárgyból igen magas a lemorzsolódók aránya minden karon.

Az előbbi probléma alapvetően egy osztályozási probléma öt osztállyal. Mivel azonban viszonylag kevés adat állt a rendelkezésünkre, ezért az adatrekordok esetén a pontos érdemjegy helyett érdemjegy csoportok prediktálásra koncentráltunk. A matematika érdemjegycsoportok prediktálására így kétféle modell került felvázolásra: egy *3 csoport modell*, illetve egy *2 csoport modell*.

A 2 csoport modell esetén a két csoportot a {5,4,3} illetve {2,1} osztályok adják, míg a 3 csoportnál az osztályok {5,4}, {3,2} illetve {1} módon alakultak. Az előbbi esetben az osztályok intuitívan a lemorzsolódási veszélyeztetettség szerint formálódtak, míg az utóbbiban egy általánosabb "jól teljesítő", "rosszul teljesítő", "lemorzsolódott" csoporthármast

[1] A szigmoid függvény: $\sigma(z) = \frac{1}{e^{-z}}$, ahol $z = x_1w_1 + x_2w_2 + \cdots + x_nw_n + b$ a súlyozott összeg.

kívántunk elérni. Mindkét modell esetén külön vizsgáltuk a teljesítményt összes hallgatóra vonatkozóan illetve szétbontva vegyészmérnök és biomérnök hallgatókra egyaránt (a környezetmérnökökről nem állt rendelkezésre elég adat, így őket a szakonkénti bontásban kihagytuk).

A két modellnél a célváltozó-értékeket rendre 1, 0-ra illetve 2, 1, 0-ra módosítottuk. Az osztályozó algoritmusok közül a Naive Bayes, kNN és Gradient Tree Boosting alapból jól kezeli a kategorikus változókat, az SVM-nél valamint a regressziós eljárásoknál viszont minimális változtatásra volt szükség. Az SVM és logisztikus regresszió mivel csak bináris osztályozásra alkalmas, így ott *One-Vs-Rest* elvű osztályozást használtunk, lineáris regressziónál pedig az címkeértékek ordinális jellege miatt egyszerűen a prediktált értéket kerekítettük a legközelebbi csoportcímkére.

Az egyes modellek és almodellek esetén főkomponens analízist is alkalmaztunk, amelynek az a lényege, hogy a rendelkezésre álló változókból kevesebb, új változókat hozunk létre miközben arra törekszünk, hogy a folyamat során elveszített információmennyiség minimális legyen. Ennek előnye, hogy az algoritmusok gyorsabban tanulnak és sokszor jobb eredményt érnek el.

A kumulált átlag prediktálásánál csak lineáris regressziót használtuk és annak a vizsgálatára összpontosítottunk, hogy az egyes években a vázolt regressziós modelleknél melyek a legfontosabb változók, és hogy ezek milyen mértékben és irányban változtatják a predikciót.

## 5.3. Implementálás és optimalizálás

A modellezési fázis megkezdése előtt a folytonos változók kvantilis alapú 0-1 skálázásnak lettek alávetve, hogy a távolság alapú osztályozók jobban teljesítsenek. Fontosabb optimalizációs lépések csak a 2-3 csoport modellek esetén történtek. A Naive Bayes és lineáris regresszió algoritmusoknál jellegük és/vagy implementálásuk végett nem történt optimalizálás. A többi algoritmus esetén az alábbi hiperparaméterek kerültek változtatásra:

- **kNN**:
  - Távolságmérték (euklédeszi, Mahalanobis)
  - Szomszédok száma (5 és 15 között változtatva)
  - Szomszéd címkeértékének súlyozása (uniform, távolság reciprokra, távolság reciproknégyzete)

- **Gradient Tree Boosting**:
  - Tanulóráta (0.01 és 5 között változtatva 0.1-es lépésközzel)
  - Boosting fázisok száma (5 és 100 között változtatva 5-ös lépésközzel)
  - Vágási feltétel (négyzetes hiba, Friedman MSE)
  - Maximális famélység (3,4 és 5)

- **Logisztikus regresszió**:
  - Regularizációs paraméter (0.05 és 5 között változtatva 0.05-ös lépésközzel)
  - Optimalizálási módszer ("SAG", "SAGA")

- **SVM**:
  - Regularizációs paraméter (0.1 és 5 között változtatva 0.1-es lépésközzel)
  - Magfüggvény (lineáris, legfeljebb 3-ad fokú polinomiális, RBF)

A PCA esetében a főkomponensek számát 1 és 8 között változtattuk, és az algoritmusokat minden főkomponensszám mellett 5-szörös keresztvalidációval optimalizáltuk, majd a legjobb modelleket kiértékeltük a teszthalmazon.

# 6. Modellek kiértékelése

# Hivatkozások