EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

---

Bartalis Dávid

# Adattömörítés szubmoduláris kiválasztással

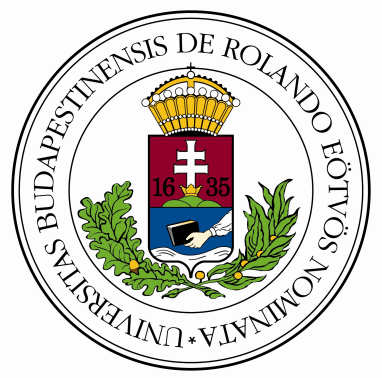
Témavezetők:

Bérczi-Kovács Erika

ELTE, Operációkutatási Tanszék

Béres Ferenc

SZTAKI, Informatikai Kutatólaboratórium




**Budapest, 2021**

# 1. Bevezetés

Már előző félévben is a szubmoduláris függvényekkel, ezenbelül a gépi tanulásban, a tanítási folyamatok felgyorsításában játszott szerepükkel foglalkoztam. Témavezetőimmel ezen témát folytattuk és egy Python csomagot (Apricot) tanulmányoztunk, újabb adathalmazokon próbáltunk ki. Azt vizsgáltuk, hogy vajon milyen adatok, kiértékelési metrikák esetén hasznosak az itt implementált függvények, illetve milyen feladatok megoldására (sorok, oszlopok redukálására) tudjuk őket használni. A továbbiakban érdekes feladat lenne egy új szubmoduláris függvény implementálása is.
Beszámolómat egy elméleti összefoglalóval kezdem, majd a vizsgált adatokról, elért eredményekről, további célkitűzésekről esik szó.

# 2. Szubmoduláris függvények - Apricot

**2.0.1. Definíció** (Szubmoduláris függvények)**.** *Egy $\mathcal{F} : 2^V \to \mathbb{R}$ $V$ alaphalmaz részhalmazain értelmezett halmazfüggvényt pontosan akkor nevezünk szubmodulárisnak, vagy teljesen szubmodulárisnak, ha $\forall B \subseteq A \subseteq V$ halmazokra és $x \in \overline{\mathrm{A}}$ esetén igaz, hogy*

$$\mathcal{F}(A \cup x) - \mathcal{F}(A) \leq \mathcal{F}(B \cup x) - \mathcal{F}(B)$$

Természetesen másként is lehet definiálni a szubmoduláris függvényeket, viszont a mi céljainkhoz ez a leghasznosabb, mert ez tükrözi a legjobban azt, hogy miért is szokták a szubmodularitást a csökkenő hasznok elveként is emlegetni. Az Apricot program egy lehetőséget biztosít hatalmas adathalmazok redukálására úgy, hogy az így kapott kisebb méretű adathalmaz reprezentatív legyen. Ezt a gépi tanulási folyamatok esetleges felgyorsítására lehet felhasználni oly módon, hogy a tanítási adathalmazt csökkentjük az Apricot programban implementált függvények segítségével, majd ezen a már jóval kevesebb sorból álló adathalmazon tanítjuk a modellünket. Ha a használt függvényt jól választjuk meg, akkor az így betanított modell eredményessége nem romlik számottevően.
A programban egyre több függvény van implementálva. Azon három függvény, melyekkel a saját, illetve a szakirodalomban fellelhető tapasztalatok alapján a legjobb eredményt lehet elérni, a következő:

- Feature-based / Tulajdonság alapú függvény:

$$\mathcal{F}(X) = \sum_{d=1}^{D} w_d \phi \left( \sum_{x \in X} m_d(x) \right)$$

- Facility location / Szolgáltató elhelyezési függvény:

$$\mathcal{F}(X) = \sum_{v \in V} \max_{x \in X} \delta(x, v),$$

- Max Coverage / Maximális fedés függvény:

$$\mathcal{F}(X) = \sum_{i=1}^{d} \left( \left( \sum_{x \in X} x_i \right) > 0 \right)$$

Ezen függvények mindegyike teljesíti a szubmodularitás definícióját; előző félévi beszámolómban ezekről részletesebben értekeztem.

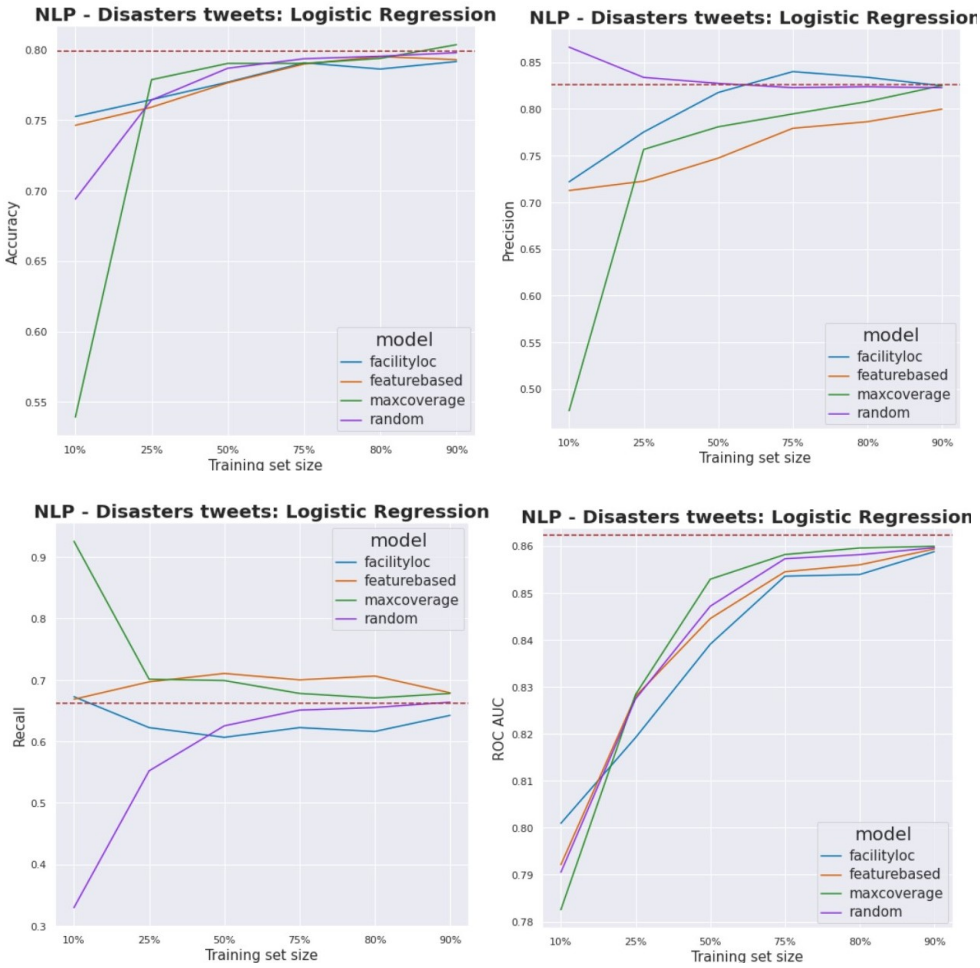
## 3. Saját eredmények

A tavalyi évben használt adathalmaz (Titanic: Machine Learning from Disaster) a kis mérete, illetve a redundancia hiánya miatt nem sok információt árult el a feltérképezendő program működéséről, hatékonyságáról, ezért ezen félévben témavezetőimmel azt tűztük ki fontos célnak, hogy nagyobb, lényegesen redundánsabb adathalmazokon is megvizsgáljuk, hogy teljesít az Apricot. Az adathalmazokat idén is a Kaggle oldalról választottuk. A legfontosabb és minden notebookban használt függvényeket feltöltöttük Githubra és innen importáltuk be a kaggle notebookokba. A mérések rögzítése a neptune segítségével történt.

### 3.1. Disaster Tweets (NLP feladat)

Az első adathalmazzal egy NLP (Natural Language Processing) feladatot szerettünk volna megoldani. Az adat különböző tweetekből állt, melyekről el szeretnénk dönteni, hogy valójában egy igazi katasztrófáról szólnak-e, vagy nem. Tehát lényegében ez alapján szeretnénk klasszifikálni a tweeteket.

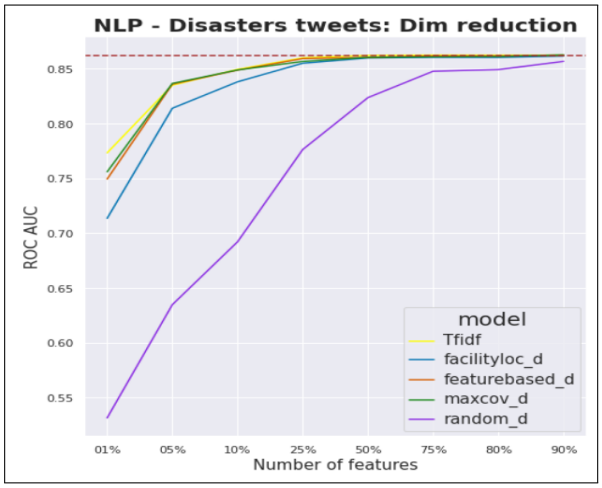
Az adathalmaz beágyazása, tisztítása után random train-test vágást használva a training adathalmaz 5264 sorból és 10000 oszlopból állt. Először a sorok számát próbáltuk csökkenteni az Apricot programot használva 10, 25, 50, 75, 80, 90 százalékra, majd ezen tanítottuk a modellt, mely először Logistic Regression volt. A redukcióhoz a fentebb definiált függvényeket, illetve random kiválasztást használtunk (ez utóbbit többször, majd kiátlagoltuk az eredményeket). A kapott értékeket a következő ábrák szemléltetik:




Az ábrákról leolvasható, hogy ha az adathalmazunk sorait 10%-ra csökkentjük, akkor jelentősen romlik az elért eredmény, viszont a sorok számát felére csökkentve már nem olyan lényeges a különbség, az így betanított, illetve az egész adaton tanított modell között. Ami szintén szembetűnő, hogy sajnos az Apricot nem sokkal múlja felül a random módszert, olykor egyáltalán nem.

Ezután megpróbáltuk más modellek tanítására használni a módszert, mint például a Gradient Boosting Classifier, vagy a Random Forest Classifier. A Gradient Boosting Classifier esetében először belefutottam abba a hibába, hogy nem volt eléggé jól paraméterezve a modell, ezért nagymértékben túltanult. Ennek következtében, amikor az Apricot módszerrel csökkentettük a sorok számát, javulást tapasztaltunk a modell jóságát tekintve, viszont a modell további hiperparaméterezése után ez a jelenség már nem volt jelen. Összességében az mondható el, hogy a sorok redukciójával nem értük el a várt eredményt, ezért más irányban próbálkoztunk tovább.

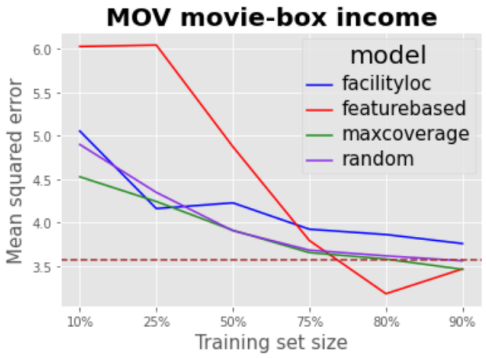
Arra gondoltunk, hogy a sorok csökkentése helyett használhatnánk ezen szubmoduláris függvényeket az oszlopok számának csökkentésére, azaz dimenzióredukcióra is. Ehhez az adathalmazt tartalmazó mátrixot transzponáltuk, majd az így kapott mátrix sorait az Apricot módszerrel csökkentve kapott mátrixot visszatranszponáltuk. Végül ezen tanítottuk a modellt. A Logistic Regression modellt és a ROC értéket tekintve a következő eredményt értük el:

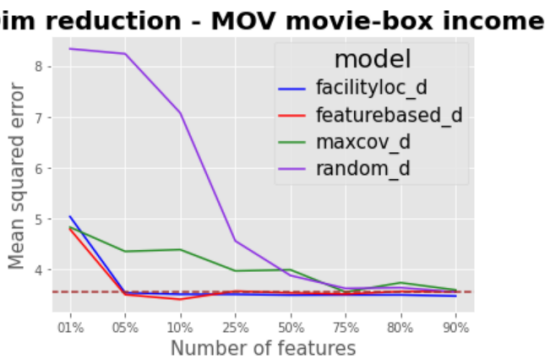



A grafikonról leolvasható, hogy a 10000 oszlopot már 10%-ra csökkentve is alig romlik a modell. Azt is fontos hangsúlyozni, hogy ebben az esetben a szubmoduláris kiválasztás (bármely függvényt használva) sokkal jobb eredményt produkál, mint a random szelekció. A szubmoduláris megközelítés mellett egy már jól bevált feature selection technikát is használtunk, azaz megvizsgáltuk az egyes szavaknak a Tfidf (term frequency–inverse document frequency) értékét és azon szavakon tanítottuk a modellt, melyekre ez az érték a lehető legnagyobb. Ezen módszerrel az Apricot-hoz hasonló eredmény született. (Az ábrán ezt a sárga vonal jelzi.) Úgy gondoltuk, hogy hasznos lehet ebben az irányban folytatni a kutatást és esetleg olyan adathalmazokon kipróbálni az oszlopok redukálását, ahol nem tudjuk a Tfidf módszert használni.

### 3.2. MOV movie-box income

A következő vizsgált adathalmaz 7000 rekordja mind egy-egy filmről szóló ismereteket tartalmazta. A feladat egy adott filmhez a bevétel megjóslása, amelyhez a lightGBM boosting módszer bizonyult a legjobbnak. Kipróbáltuk az adathalmazon a sorok redukálását és dimenziócsökkentést is a szubmoduláris függvények segítségével. Az így elért eredmények az alábbi ábrákon láthatók. Érdekesség, hogy úgymond dupla redukcióval is próbálkoztunk, azaz először csökkentettük az oszlopok számát, majd a sorok számát is, természetesen az Apricot csomagban implementált függvényekkel, viszont így nagy mértékben romlott az elért erdmény.




## 4. Összegzés, további tervek

A félév során újabb modelleket (LightGBM, XGBoost, ...) ismertem meg, sok implementációs trükköt tanultam. Az implementáció során mindig figyelni kellett arra, hogy az adatok típusa megfelelő legyen, ugyanis az Apricot által használt függvények erre érzékenyek.

A mérések során azt tapasztaltuk, hogy a dimenziócsökkentés témakörével célszerű tovább foglalkozni és ezen tesztelni a szubmoduláris szelekciót. A szakirodalomban írnak arról, hogy a szubmoduláris feature redukció pénzügyi adatokon kifejezetten jól működik, ezért rövid távú célként azt tűztük ki, hogy ilyen adathalmazzal is kísérletezzünk. Mint már korábban is említettem a fő célunk az lenne, hogy megtaláljuk, mely feladatra használható valóban rendkívül jól ez a módszer, majd ezután egy saját szubmoduláris függvényt implementálnánk az Apricot csomagba.