Tapasztalat beszámoló

Bevezetés

- Bevezetésként meghallgattunk egy részletes beszámolót a projektünkről, illetve az általunk használt technológiákról és az architektúráról.
- Megterveztük a nyári gyakorlat ütemét, részletesen annak elejét:
    - Szakirány választás az elején, magam az modellezést és a *Python* alkalmazások fejlesztését választottam
    - Első két hétben egy mini-projektek kell elkészítenünk, amiben minden technológiát kell alkalmazni, segítségével végén el tudjuk dönteni, hogy tényleg ezt akarjuk-e.

Első két hét eredménye : mini-projekt

Feladat

- Bármilyen front-end technológiával adatok vizualizálása *SAP Hana* adatbázisból nyert adatokkal *ODATA* segítségével.

Előkészületek

- Sikeresen létrehoztam egy *SAP Hana Trial* fiókot, amivel használni tudtam a SAP Web IDE-t, először *SAP* szerveren, majd technikai problémák miatt később saját szerveren.
- Még mielőtt nekikezdtem volna a teszt projekt létrehozásának, előtte több napig oktatóanyagokat olvastam magáról a környezetről, az architektúráról (https://help.sap.com/viewer/4505d0bdaf4948449b7f7379d24d0f0d/2.0.03/en-US/d8226e641a124b629b0e8f7c111cd1ae.html). Tudatában, hogy a projektben adatbázisra van szükségünk, eddigi tudásaimat kiegészítettem ebben a

témakörben (http://scs.web.elte.hu/Work/DW/adattarhazak.htm). Ezáltal elsajátítottam rengeteg fogalmat, mint például az adatkocka, pivot, delta adattöltés jelentéseit.

- Technikai problémák miatt nem tudtuk használni a *SAP Web IDE*-t, illetve még ekkor nem tudtuk, hogy milyen témakörben fogunk dolgozni, ezért további ismeretek elsajátításába kezdtem:
    - Először is egy *Python* gyorstalpalót kezdtem, pár nap alatt sikerült segédanyag mellett megtanulni, illetve begyakorolni (https://www.w3schools.com/python/python_intro.asp).
    - Ezután kicsit front-end technológiák iránt kezdtem el érdeklődni, például a *React* (https://www.w3schools.com/react/), és az *Angular* (https://www.w3schools.com/angular/) után. Ezeket a tudásokat nem sajátítottam el, inkább csak olvasgattam róluk.
    - Legutolsósorban a *REST Api* (https://en.wikipedia.org/wiki/Representational_state_transfer) és az *ODATA* elmélete (https://www.odata.org/getting-started/understand-odata-in-6-steps/), iránt érdeklődtem.

Mini-Projekt megkezdése

- Második hét elejére végre elkészült a saját szerverünk, így meg is kezdődött a saját mini-projektünk elkészítése.
- Először is létrehoztam egy új *MTA applikációt* a *SAP Web IDE Workspace*-ben, ami egy webes felületet jelenít meg, míg perzisztencia rétegét adatbázis támogatja. Továbbá fontos tulajdonsága, hogy az applikáción belül létre tudunk hozni kalkulációs nézetet, ami később igen fontos lesz.

Adatbázis létrehozása

- Projekten belül létrehoztam egy *SAP HANA Database Module*-t, ami alapján csatlakozni tudunk adatbázishoz, illetve ezen belül adatbázishoz szükséges kellékeket (kontextus, tábla, nézet) tudunk létrehozni. Ezt build-elve automatikusan létrehozott, illetve csatlakozott a saját adatbázisunkhoz (ha esetleg nem automatikusan, akkor *Database Explorer*-ben manuálisan is megtehetjük).
- Az előbb készített mappában ezután létrehoztam egy *HDB CDS Artifact*-ot, ami egy adatbázis kontextus, itt tudjuk definiálni a szükséges fogalmakat (típusokat) a táblákhoz, illetve létre tudunk hozni táblákat, azoknak attribútumjait, illetve megszorításokat tenni rá. Jelen esetben létrehoztam az általunk megkapott tábla sémáját, illetve megszorításait (kulcs például), aztán magába a táblába beimportáltam az adatokat.
- Ezután megkezdődött a normalizálás és a tisztítás: a normalizált táblák sémáját szintén létrehoztam, a fő táblából feltöltöttem ezeket ügyelve arra, hogy kiszűrjem a duplikátumokat, illetve a hibás adatokat javítsam, de a forrás rengeteg elírást tartalmazott, emiatt nem sikerült eléggé tisztítani.

Kalkulációs nézet létrehozása

- Először is utánanéztem a kalkulációs nézetnek, pontosan mit is jelent ez, illetve milyen fajtái vannak, illetve mik is az előnyeik (https://www.guru99.com/sap-hana-calculation-view.html). Lényegében egy nézet, amiben hatékonyabbak a lekérdezések és aggregációk.
- Majd megkezdtem a létrehozását, először is a *HDB CFS Artifact*-ban létrehoztam egy *SQL Access Only* kalkulációs nézetet, join típusként projekciót használtam, ugyanis szimplán az adatokat akartam elérni. Ha

bármilyen aggregációt használtam volna, akkor természetesen *CUBE* típusút, illetve a aggregáció join-t használtam volna, mivel úgy hatékonyabb.

- Ezután hozzáadtam még egy join-t, amiben megadtam hogy a normalizált táblákat hogyan kösse össze egymással, illetve a join output-jához hozzáadtam minden táblát, ugyanis minden adatra kíváncsiak vagyunk. Majd végül ezt rákötöttem a projekciós join-ra, és ott szintén minden adatot hozzáadtam output-nak.
- Ezután egy build volt szükséges a sikeres kalkulációs nézet elkészüléséhez.
- Az elkészítés nem hivatalos tutorial alapján készült, ugyanis azok lényegében eltértek a mi adatbázisunktól, illetve az a verzió sokkal régebbi volt. Csoportban talán az elsőnek sikerült nekem, ezért sok időt szánva többieknek írtam egy tutorialt, illetve személyesen segítettem nekik, egy kis magyarázattal kiegészítve.

Összekapcsolás *ODATA*-val

- A cél az volt, hogy egy *Node.js* modult futtatva url-en keresztül külsőleg le lehessen kérdezni adatokat *ODATA* segítségével.
- Emiatt létrehoztam egy új *Node.js* modult az *MTA* applikációmban és a következőket módosítottam:
    - lib mappában létrehoztam egy xsodata kiterjesztésű fájlt, amiben megadtam, hogy melyik táblámat miként tudjam elérni az url-ben *(service{ <elérni_kívánt_táblanév> as <url-ben_hivatkozandó_fájlnév>; }).*
    - *server.js* fájlban a “*redirectUrl*”-t az előbb készített xsodata fájl nevét megadtam.
    - *mta.yaml* fájlban az odata modulban megadtuk hogy szüksége van a *HDI container*-re (*requieres : - <hdi_container_module_név>*)
- Ezek után futtatva sikeresen elérjük a táblát külön lekérdezés szintaktikával url-en keresztül, ezáltal bármilyen front-end technológiával el tudjuk érni nézettábla adatait.

Adatok megjelenítése *Angular*-ral

- Első hét alatt szereztem minimális *Angular* ismereteket, ezt bővítettem tovább mielőtt nekikezdtem volna a megjelenítésnek tutorial-ok alapján

(https://www.npmjs.com/package/angular-odata-es5). Összefoglalva megismertem, hogy hogyan épül fel egy *Angular* applikáció, milyen komponensei vannak, illetve hogyan lehet ezeket használni. Itt beláttam, hogy nem szeretnék front-end technológiával foglalkozni a későbbiekben, ugyanis annyira nem tetszett meg.

- Még mielőtt nekikezdtem volna a vizualizáció megírásához, először is a projektnek hozzá kell tudnia csatlakozni az előbb elkészített back-end szerverünkhöz, ezt *proxy* segítségével sikerült megoldani egy tutorial alapján (https://medium.com/bb-tutorials-and-thoughts/angular-how-to-proxy-to-backend-server-6fb37ef0d025). *Proxy* segítsége nélkül ugyanis nem tudtuk volna direkt elérni a *SAP Web IDE*-ben futtatott *Node.js* modul által generált url-t
- Sikeres csatlakozás után további *Angular* ismeretek szereztem, egy tutorial (https://www.codementor.io/antonselin/consume-odata-service-restful-angular-2-uj1i6szsw), és az eddig tanult ismeretek segítségével el tudtam készíteni a saját projektet, aminek az eredménye az adatok diagramos ábrázolása, szűrhető feltétellel (aminek az értékei szintén az adatbázisból vannak). Miután kész lett letisztítottam a kódot, illetve megvalósítottam úgy, hogy később tovább bővíthető legyen, de idő hiányában, illetve a szakosodás miatt erre már nem jutott sor.

Második két hét eredménye: adatbányászati tudás részleges elsajátítása

Bevezetés a témakörbe:

- Második két hétben megtörtént a szakosodás: nekem az adatbányászat jutott Hana környezetben, azon belül a *PAL* algoritmusok. Első két hétben ezekkel már találkoztam, de ezekhez mély elméleti ismeret szükséges, úgyhogy rengeteg ismeretre, ami még így sem teljes, tettem szert.
- Számomra teljesen ismeretlen volt ez a témakör, úgyhogy először is egy bevezetővel (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch01.html) kezdtem.
    - Itt lényegében megtudtam, hogy mire használható ez a tudás, az algoritmusoknak milyen fajtái vannak, melyikkel milyen új ismeretekre tudunk szert tenni.

- Mivel több algoritmus fajta is van, ezért csak a négy alap típusra szántam az időmet, mivel talán ezek a legfontosabbak. Ezek nem mások mint a klaszterezés, regresszió, osztályozás és a asszociációs analitika.

Algoritmusok elsajátítása:

- Először is az asszociációs analitikával kezdtem, ugyanis ez tűnt a legegyszerűbbnek, a hozzám legközelebb állónak.
    - Szintén kezdtem egy bevezetővel (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch06.html), aminek segítségével megértettem a használatát.
    - Ezután azokat az algoritmusokat ismertem meg, amit használni lehet *PAL* környezetben:
        - Ilyen például az *Apriori* (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch06s05.html), ami szuperhalmazok segítségével képezi a szabályokat (ha a nagyobb halmazból lehet szabályt alkotni, akkor annak részhalmaziból is).
        - Illetve az *FP-Bővítés* (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch06s06.html), ami ugyanis fát épít és ennek segítségével meg az összefüggéseket.
        - *KORD* algoritmus, ami az első *k* darab legérdekesebb elemet keresi
    - A kellő tudást megszereztem, hogy megnézzem mind ezek hogyan működnek *PAL* környezetben (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/7a073d66173a4c1589ef5fbe5bb3120f.html).
- Áttértem egy másik adatbányászati témakörre, a regresszióra:
    - Először is az elméleti hátterével foglalkoztam, hozzá kapcsolódó fogalmakat megismertem (https://www.sciencedirect.com/topics/engineering/regression-coefficient), illetve magát a használatát.

    - Ezután áttértem a *PAL* környezeti regressziókra (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/1.0.12/en-US/4b4e884a82204b5f81b506dad15ce7f7.html), megismertem milyen fajták vannak, amik ugyanazt számolják, csak más matematikai képletekkel állítják elő az egyenletet, amivel megkapjuk az eredményt.
    - Ilyenek például a *geometriai*, *logaritmikus*, *exponenciális* és *polinom* regressziók.
- Ezután viszont egy nehezebb témakörre ugrottam át: az osztályozásra:
    - Hasonlóan az előzőekhez itt is elméleti anyaggal kezdtem (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch04.html), ami itt már sokkal bővebb volt.
    - Az alapokkal kezdtem, mint például a döntési fákkal, azoknak az építésével, milyen használatuk van az osztályozásnál (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch04s02.html, https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch04s03.html). *PAL*-ban ezeket lehet használni osztályzásra, de sajnos idő és nehézség hiány ezeket nem tudtam alkalmazni.
    - Utána osztályozási módszerekkel, azon belül leginkább az összehasonlítási módszerekkel (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch04s06.html).
    - Továbbá a szabályalapú osztályzással (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch05.html). Bár ezeket *PAL*-ban nem igen tudtam hasznosítani, de plussz tudásnak nem ártott.
    - Végül legközelebbi szomszédok módszerével (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch05s02.html) foglalkoztam, megértettem, hogy valóban a k legközelebbi szomszéd alapján osztályoz és hogy mennyire veszélyes ez a *k* szám kiválasztása, ugyanis ha rosszul választunk lehet, hogy teljesen torz eredményeket kapunk.
- *PAL*-ban folytattam az osztályozást:
    - Először is a *PAL* döntési fákkal foglalkoztam (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.

03/en-US/c8ab80c6c68543c89602796860172533.html), ugyanis a PAL döntési fa 3 ismert másikból lett összegyurva, illetve kieegészítve. Majd *PAL* környezetben átfutottam az itteni fogalmakat (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/79c0d16e4b5140a59115c9c694f99a78.html).

    - Konfúziós mátrixok, azoknak használatuk az algoritmusok folyamán.
    - Ezután kezdtem el a tényleges *PAL* osztályozási algoritmusokat, mint például az *AUC*-ot (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/df08788a52744c899fab13a8712d30e8.html). Ami bár nem osztályzási algoritmus,viszont segítségével meg tudjuk állapítani, hogy mennyire precíz, mennyire működik helyesen a használt algoritmus.
    - Illetve a *K Legközelebbi Szomszéd* algoritmust (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/f2440c6b3daa41dd8cc9fc5b64805a68.html), ami *k* legközelebbi szomszéd alapján osztályoz: többség vagy távolság alapján.
    - Több algoritmust nem sikerült elsajátítani nehézségek miatt.

- Az utolsó témakört, a klaszterezést hagytam legutoljára:
    - Szokásos módon egy bevezetővel kezdtem (https://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch09.html). Mivel elég részletes volt, elég nagy anyag ugyanis, ezért elég is volt kezdésnek, megértettem ennek a típusnak a célját és hogy nagyjából hogyan fog működni, szóval rátértem a *PAL* klaszterezésre.
- *PAL* környezetben szintén rengeteg klaszterezési algoritmus található, ezért a az alap algoritmusokat tanulmányoztam:
    - Először is *K-Közép* és hozzá hasonló algoritmusokkal kezdtem (*K-Medián*, *K-Medoid*):
        - Használatuk a különböző fajtájuknak *PAL*-ban(https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/53e6908794ce4bcaa440f5c4348f3d14.html, https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/0ba6f4b13aa549af824448574628cb4a.html). Majd a gyorsított verziójával

(https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/5a3e282d46fa4b40a9a78520afff084d.html).

- Illetve maguk a működési háttér: pontok alapján klaszterek kiszámolása, pontok hozzárendelése majd ezek alapján újraszámolás.

- *Összevonó hierarchikus klaszterezés*:
  - Működése: Elején minden egyes elem egy külön klaszter és ezeket szépen távolság alapján összevonja míg egy klaszter marad csak.
  - *PAL*-ban használata (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/394a529f927b4ecc88008f0d60d61ac1.html).
- *DBSCAN*:
  - Működése először: Sűrű elemeknél érdemes alkalmazni, hiszen egy *epsilon* sugarú környezetbeli elemeket rendel a klaszterhez, már ha elég van belőlük.
  - *PAL*-ban használata, paraméterek stb (https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/b2c55113763f4fcebd31a7486daadd08.html).
- *AP:*
  - Működése: hasonlóság alapján végzi a klaszterezést és ezek alapján hozza létre a klasztereket, amik lehetnek adatpontok (ezeket lehet súlyozni), illetve független pontok is.
  - Először is egy elméleti háttér tanulmányozása(https://en.wikipedia.org/wiki/Affinity_propagation), majd *PAL*-beli használata(https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/ec478f518d0a4160ae34d62d03981cd1.html).

Utolsó két hét eredménye:

Előkészületek az összehasonlításhoz:

- Tervezett feladat utolsó két hétre az lett volna, hogy egy megkapott táblán különböző adatbányász algoritmusokat, különböző paraméterekkel futtatni és elemezni az eredményeket, illetve optimalizálni a futási időt.
- Mivel a tervezett feladat késett ezért saját feladatot kreáltam:
    - Először is sikerült megoldani, hogy hogyan tudok jogosultság hiányában *PAL* eljárásokat használni saját eljárásokban: saját *workspaceben* létrehozni ezeket a metódusokat, külön szintaxissal, majd ezeket meg lehet hívni egy rendes eljárásban.
    - Ezáltal lehetőségem volt kipróbálni a tanult algoritmusokat különböző példákon keresztül:
        - Klaszterezés: *DBSCAN*, *K-Means*, *AP*, *hierarchikus összevonó klaszterezés*
        - Osztályozás: *AUC*, *KNN*
        - Regresszió: *Logaritmikus, exponenciális, geometriai* stb.
        - Asszociáció: *FP-Growth*, *Apriori*
    - Kipróbált algoritmusok viszonylag egyszerű adatset-en történtek. Az adatok nagyrészt a hivatalos *PAL* dokumentáció példáiból jöttek, ezért azért volt hasznos, mert tudtam mit kell kapnom eredményül, illetve olyan példák voltak, amik az adott algoritmusra tekintve “látványos” eredményeket produkáltak.
    - Legutolsó sorban az algoritmusok paramétereivel próbálkoztam, kísérleteztem, sok esetben minimális eltérés történt, míg azok a paraméterek amik jelentősen befolyásolják az algoritmus működését (például *KNN* módszernél nem a *default* módon magától határozza meg az *epsilon* és a minimális pontok értéket, hanem mi adjuk meg) akkor jelentősen megváltozik a kapott eredmény.
- Ezáltal sikerült birtokba venni, hogy hogyan lehet használni ezeket az algoritmusokat, sajátot létrehozni amikben ezeket meghívjuk, illetve a felparaméterezés hogyan történik. Így készen álltam a feladatra.

PAL adatbányászat összehasonlítása *Pythonnal*:

- Utolsó feladatként megkaptuk, hogy hasonlítsuk össze a PAL és a Python környezetbeli adatbányászatot, bizonyos szempontok alapján.
- Teljesen összehasonlítani nem tudtuk a két könyvtárat, ugyanis rengeteg közös és más-más algoritmus található mindkettőben, illetve ezek is rengeteg féle képpen összehasonlíthatóak.
- Természetesen úgy lett volna a leglátványosabb, hogy több adathalmazra is elvégezzük az összehasonlítást, de idő hiányában erre nem tudtunk sort keríteni.
- Emiatt, egy jól ismert adatset-en dolgoztunk, az Irisen. Ugyanis ezen rengeteg klaszterező és osztályozó algoritmus látványosan működik.
- Az összehasonlítás összefoglalva:
    - A nulláról kezdtük, hogy ha valaki hozzá szeretne férni egyik könyvtárhoz, ahhoz először is mit kell tennie hozzá. Lényegében a hozzá szükséges dolgokat soroltuk fel, leginkább az előnyöket és a hátrányokat: például PAL-nál hamar be tud terni a tárhely egy asztali számítógépen, de szerverről használva rengeteg pozitívum található.
    - Utána azzal folytattuk, hogy hogyan lehet használni az adatbányász eljárásokat, miket kell hozzá előre megcsinálni.
        - Python-nál szimplán megadjuk a forrás fájlt (esetleg adatbázishoz csatlakozunk) és a megfelelő utasításokat futtatva el is készültünk. Ezeket az eljárásokat szerencsére tudjuk tárolni.
        - Addig PAL-ban létre kell hozni a külön sémákat manuálisan, egyesével feltölteni, segéd táblák sémáit a dokumentációból kinézni egyesével, ami néha elavult tud lenni, tehát sokkal időigényesebb ezt használni sajnos. Viszont el tudjuk tárolni az eljárásokat az adatbázisba könnyedén, ezáltal újrafelhasználható lesz és mások is tudják használni, akik az adatbázishoz csatlakoznak. (Természetesen ez Python környezetnél is megvalósítható, de nyilván bonyolultabb ez a része, hogy több mindenki tudja használni).
    - Még az algoritmusok paramétereit vizsgáltuk, egyes algoritmusok mennyire testreszabhatóak. Azt a következtetést sikerült levonni, hogy

mindkét környezetben viszonylag módosíthatóak minimálisan szinten az algoritmusok (például távolságok kiszámítása, szálak eloszlása, keresési módszerek), de Python-ban jobban lehet befolyásolni azokat a részeket, ami jelentősen megváltoztatja az eljárást és ennek függvényében az eredményt.

    - Legutolsó sorban az eredményeket vizsgáltuk. Viszonylag hasonló paraméterekkel a futási időt, illetve a kapott eredményeket. Itt megjegyezném, hogy a PAL rengeteg hasznos outputot ad, kezdve a várt eredményeket, de adhat statisztikát, pontosságot akár, ami későbbi felhasználáshoz hasznos tud lenni. Viszont vizualizálni az adatokat már egyre kevésbé, elég bonyolult és időigényes módszerrel, amire nekünk már nem jutott idő. Ezzel szemben Python-ban nagyon látványos tudja megjeleníteni az eredményeket.
    - Kicsit összegezve az elemzést, sikerült nagyjából feltárni a PAL és Python környezetek előnyeit és hátrányait. Eredményt tekintetében elég hasonlóak a működésük. Nincs egyértelműen jobb, mindkettő más-más miatt jobb mint a másik.
    - Még utolsó sorban megjegyezném, hogy PAL-ban rengeteg olyan problémával találkoztam, amik igen bosszantóak tudnak lenni, de valószínűleg ez még a kiforratlanságának tudható be, ami talán később javulni fog.

- A részleges, precíz összehasonlítás megtalálható a szakmai gyakorlat git repository-jában

Refaktorálás és naplózás

- A megbeszéltek alapján, miután mindenki végzett a saját feladatával, ami az esetemben egy összehasonlító elemzés, a maradék időtartamban refaktoráljuk az eddigi munkát, illetve dokumentálunk. Abban az esetben, ha nem végeztünk volna, ekkor lett volna idő befejezni a kiszabott munkát.
- Szerencsére elkészítettük az elemzést, így a maradék időszakban az alábbiakat tettem refaktorálás szempontjából:
    - Először is az adatbányász összehasonlító elemzést újra elolvastam, esetleges hibákat, pontatlanságokat javítottam ki, illetve, ha hiányzott valami belőle akkor azt pótoltam.
    - Közös munkánkat szebbé tettük formázással, szebb elrendezést kialakítva, illetve ábrák segítségével látványosabbá tettük.
    - Ezzel el is készültünk teljesen az elemzéssel.
- Hátramaradt a naplózás, amit szerencsére egy vázlatos napló alapján könnyen sikerült rekonstruálni a történteket, ugyanis azokat a pontokat kellett egybefüggő szöveggé alkotni, néha kiszűrni a lényeget, majd formázni.