Kodolányi János Egyetem

SZAKDOLGOZAT

Jónás Attila
Gazdálkodási és menedzsment alapszak

Budapest
2022

Kodolányi János Egyetem
Gazdálkodási és Menedzsment Tanszék

Nyílt adatbázisok automatizált elemzése OAM megközelítés alkalmazásával

Konzulens: Dr. Pitlik László

Készítette: Jónás Attila
Gazdálkodási és menedzsment alapszak

Budapest
2022

TARTALOMJEGYZÉK

ÁBRAJEGYZÉK

RÖVIDÍTÉSEK JEGYZÉKE

API - Application Programming Interface - Alkalmazásprogramozási interfész

ASCII - American Standard Code for Information Interchange - Amerikai szabványos információcsere-kód

COCO - Component-based Object Comparison for Objectivity - Komponensalapú objektum-összehasonlítás az objektivitásért

CSV - Comma Separated Values - vesszővel elválasztott értékek

HTTP - HyperText Transfer Protocol - hipertext átviteli protokol

HTML - HyperText Markup Language - Hipertext jelölőnyelv

JSON - JavaScript Object Notation - JavaScript objektumjelölés

KEI - Key Short-Term Economic Indicators - rövid távú gazdasági mutatók

MEI - Main Economic Indicators - Főbb gazdasági mutatók

OAM - Objektum-Attribútum Mátrix

OECD - Organisation for Economic Co-operation and Development - Gazdasági Együttműködési és Fejlesztési Szervezet

PEP - Python Enhancement Proposals - Python fejlesztési javaslatok

SDMX - Statistical Data and Metadata eXchange - statisztikai adatok és metaadatok cseréje

TCP - Transmission Control Protocol - átviteli vezérlő protokoll

TDD - Test Driven Development - Tesztvezérelt fejlesztés

TSV - TAB Separated Values - Tabulátorral elválasztott értékek

URL - Uniform Resource Locator - egységes erőforrás-helymeghatározó

XML - eXtensible Markup Language - kiterjeszthető jelölőnyelv

1 BEVEZETÉS

Dolgozatom alapját az Egyetem, Gazdálkodási és Menedzsment szak, Digital Economy szakirányon megvalósított különálló projektje[1] adja. Ezen projekt keretében, egy önszervezett Hallgatói munkacsoport keretében megvalósítottuk (2021.12.-2022.11.15-ig) nyílt, internetes adatbázisból származó adatok -OECD[2] és DBnomics[3] adatbázisok- megadott szűrési feltételek szerinti feldolgozását. Majd ezen idősoros adathalmaz szűrése és az adatok tisztítása után az OAM modell szerinti értékbecslések alapján kalkulált eredményeket hasonlítottuk össze a tény értékekkel országok, mint objektumok és mutatószámok, mint attribútumok 10-10 havi időszakában. Mindez annak érdekében történt, hogy ki lehessen mutatni mutatószámonként numerikusan és lehessen érzékeltetni grafikusan is, vajon inkább az átlagpolgár terhére vagy előnyére történtek-e változások a mindenkori politikai aktőrök (pl. kormányok) munkájának eredményeként.

A kockázatértékelés lényege egyszerű: ha egyetlen egy mutatószám kivételével minden más adatot ténynek tekintünk minden ország minden időszakára, akkor a lehető legrugalmasabb modellezési logika mellett milyen becslési értéket kapunk a vizsgált mutatóra országonként és időszakonként? Ha a tény-becslés összevetés közel nullahibás, vagy a mutatószám logikája alapján az átlagpolgár előnyére tér el, akkor ennek illene (vö. hipotézis) kormányt stabilizáló hatással bírnia, míg fordított esetben kormányváltás lenne elvárható a közeljövőben.

A hipotézisünk bizonyítására (miszerint a gazdasági mutatók alapján megbecsülhető a politikai erők elmozdulása) jött létre a projektünk, először egyféle „brainstorming”[4], majd tényleges tervezés jelleggel. Mint már utaltam rá, a projekt sikerrel zárult, a hipotézisünk felvetésére a válasz röviden a lehetséges megbecsülni! Ez persze egy több hónapig tartó munka eredménye volt.

[1] Robot-polgár (e-gazság projekt 2022 vs 2010) (2022) https://miau.my-x.hu/miau/285/e_gazsag_2022/ (Utolsó letöltés: 2022.11.15.)

[2] Organisation for Economic Co-operation and Development – OECD (2022) https://www.oecd.org/ (Utolsó letöltés: 2022.11.15.)

[3] DBnomics – the world’s economic database (2022) https://db.nomics.world/ (Utolsó letöltés: 2022.11.15.)

[4] Brainstorming (2022) https://zsidai.com/brainstorming-jelentese.html (Utolsó letöltés: 2022.11.15.)

A projekt problémájának felvetése során a komplexebb célunk nem egyszerűen tetszőleges számok feldolgozása volt, hanem a publikus adatok alapján a politikai hatalom gazdasági hatások általi megváltozásának az esélymeghatározása. Ez már a nyers számok és komplex adatbázisok feldolgozása miatt is izgalmas kihívást jelentett, ami a gazdasági hatások és azok összefüggésének elemzésén át gyakorlatilag korlátlan lehetőséget nyitott meg a lehetséges adatelemzések tekintetében. Ennek a folyamatnak pedig jelentős része automatizálható is.

Manapság a legkisebb cégek számára is fontos az adatainak kiértékelése, tőlük pedig jelentősen nagyobb volumenben vizsgálva akár egy ország, vagy egy több országból álló szövetség is folyamatosan kell, hogy vizsgálja a rendelkezésre álló adatvagyonát. A gazdasági életben ez elengedhetetlen a versenyképesség megtartása érdekében. Ennek része a meglévő adatok alapján becsült értékek meghatározása is, hiszen a piaci mozgások súlyos veszteségeket, vagy épp hatalmas nyereségeket is okozhatnak.

Dolgozatom írása közben a fejlesztett kódhoz az igen elterjed és könnyen értelmezhető Python [5] programnyelvet használtam fel, melyhez számos, hasznos fejlesztői könyvtár érhető el, jelentősen megkönnyítve a komplex feladatok egyszerű megvalósítását.

Dolgozatommal célom a felhasznált és felépített logika bemutatása, hogy ne csak mély informatikai szakmai ismeretekkel rendelkező olvasó számára legyen értelmezhető a megvalósított program.

A Ko-Do-Libri kódnevű projektben részt vevő hallgatótársaim közül többen is a projekt különböző munkafázisát dolgoztuk fel szakdolgozat formájában, miközben már a projekt kapcsán is igyekeztünk a manapság elterjedt és alkalmazott agilis módszertanokat integrálni a megvalósításba.

A dolgozatom szerkezetével kapcsolatban:

- Dolgozatom rendhagyó a szak többi szakdolgozataihoz képest, lévén egy forráskód is a dolgozat mellé önálló munkaként megírásra került.
- A dolgozatom ennek megfelelően a terjedelmi korlátok miatt szűk keretek közé szorítva kezel minden olyan részletet, ami nem a projekt és nem a forráskód értelmezésének szerves része. Ezáltal a forráskód (mely a

[5] Python (2022) https://www.python.org/ (Utolsó letöltés: 2022.11.15.)

program prototípusának a működőképes háttere) a dolgozat értékének a szerves részét képezi.

- A dolgozat készítése során a szakirodalmi hivatkozásokat nagyrészt lábjegyzetek formájában fogom csatolni, a legfontosabbakat kiemelem a dolgozat végén is.
- A dolgozat mivel kevés külső forrásra épít, így minimális idézettel operál, ezzel is hangsúlyozva a saját munka jelentőségét a megvalósításában.

2 A FELHASZNÁLT ADATBÁZISOK

A bemeneti adatok automatizált elérése nélkül megvalósíthatatlan lenne a kiértékelés, így az elsődleges és egyben legfontosabb lépés volt feltérképezni az adatokat és azok elérhetőségét. Ebben a fejezetben erről lesz szó részletesebben.

2.1 Az OECD KEI adatok

„Together, we create better policies for better lives" - OECD

A Gazdasági Együttműködés és Fejlesztés Szervezete (Organisation for Economic Co-operation and Development, OECD) számos tevékenysége közül az egyik legfontosabb, hogy egy világméretű adatbankot üzemeltet, mely adatbázis szabadon hozzáférhető bárki számára. A szervezet által megvalósított táblázatok és kiértékelések[6] mellett a nyersadatok[7] is elérhetőek.

A projektünk az OECD-nél elérhető fő gazdasági adatok[8] közül a kiemelt, rövid távú gazdasági mutatókra, azaz a KEI (Key Short-Term Economic Indicators[9]) adatokra fókuszált. Ez az adatkészlet havi, negyedéves és éves, elsősorban növekedési rátákat és szinteket mutat meg. Többek között ilyen a negyedéves nemzeti számlák, az üzleti felmérések, az ipari termelés, a gépjármű-regisztrációk, az építőipar, a fogyasztói és termelői árak, a teljes foglalkoztatás, a munkanélküliségi ráta, a kamatlábak, a monetáris aggregátumok és belföldi pénzügyek, a külföldi pénzügyek, a kiskereskedelem és külkereskedelem, az árfolyamok vagy akár a fizetési mérleg is[10]. Ezen adatok felhasználása máris mutat egy továbbfejlesztési lehetőséget, hisz a felhasznált mutatók jelentősen finomíthatják, pontosíthatják a végeredményeket, így más mutatókkal árnyaltabb képet kaphatunk.

[6] OECD Data (2022) https://data.oecd.org/ (Utolsó letöltés: 2022.11.15.)
[7] OECD Stat (2022) https://stats.oecd.org/ (Utolsó letöltés: 2022.11.15.)
[8] Main Economic Indicators (MEI), rendszeres, havonta megjelenő kiadványok (2022) https://www.oecd-ilibrary.org/economics/main-economic-indicators/volume-2022/issue-10_c59d28e2-en (Utolsó letöltés: 2022.11.15.)
[9] OECD KEI Statistics (2022) https://stats.oecd.org/index.aspx?DatasetCode=KEI (Utolsó letöltés: 2022.11.15.)
[10] Key short-term indicators (2022) https://www.oecd-ilibrary.org/economics/data/main-economic-indicators/key-short-term-indicators_data-00039-en (Utolsó letöltés: 2022.11.15.)

Az adatok kinyerésére a https://stats.oecd.org oldalon keresztül rögtön biztosítanak számunkra egy online felületet, mely exportálási funkcióval is rendelkezik. Itt az általában megszokott offline formátumok (CSV és a Excel file) mellett egy speciális PC Axis formátumot is támogat. Továbbá elérhető már ezen felületen keresztül is a fejlesztői alkalmazásprogramozási csatoló (Developer API[11]) és külön statisztikai adatok lekérését biztosító SDMX-XML[12] formátum is.

Mivel a megvalósított programban a CSV file támogatását is elérhetővé tettem, így ennek a formátumát (fejlécek és adatsorok) definiáltam alapértelmezettnek és erre konvertáltam át a más módon is elért adatokat. Erről részletesebben később.

Az OECD adatok elérésére számos lehetőség áll rendelkezésre, melyek közül az egyik legáltalánosabbnak a PandaSDMX[13] OECD adatforrás támogatása tekinthető. Ezt használtam fel a megvalósítás során is. Ennek segítségével az előzőleg említett SDMX adatkapcsolatot elérve egy Python Pandas[14] könyvtár, DataFrame típusú, rugalmasan használható objektumába tölthetjük be az adatokat.

2.2 A DB Nomics OECD KEI adatai

„DB Nomics - the worlds's economic database" – db.nomics.world

Az OECD által biztosított felület kapcsán számos alkalommal tapasztaltam a tesztelések során sebességcsökkenést, vagy akár az adatok ideiglenesen elérhetetlenek voltak. Többek között emiatt is létezik az itt elérhető adatokból más adatbázisba is szinkronizált adattár. Az egyik ilyen a DB Nomics projekt által biztosított[15].

Itt a https://db.nomics.world/OECD/KEI oldalon keresztül egy még egyszerűbben kezelhető és a jelentősen gyorsabb felület áll a rendelkezésünkre az OECD-nél elérhető adatok lekérésére.

[11] OECD data for developers (2022) https://data.oecd.org/api/ (Utolsó letöltés: 2022.11.15.)
[12] SDMX (2022) https://sdmx.org/ (Utolsó letöltés: 2022.11.15.)
[13] Statistical Data and Metadata eXchange (SDMX) for the Python data ecosystem (2022) https://pandasdmx.readthedocs.io/ (Utolsó letöltés: 2022.11.15.)
[14] Python Pandas (2022) https://pandas.pydata.org/ (Utolsó letöltés: 2022.11.15.)
[15] DB Nomics OECD KEI (2022) https://db.nomics.world/OECD/KEI (Utolsó letöltés: 2022.11.15.)

Mivel itt szinkronizált adatokról van szó, így előfordulhat minimális eltérés azokban, emiatt minden aloldal felső részén elérhetővé teszik az utolsó frissítés időpontját.

Az adatok letöltésére itt is elérhető a CSV és az Excel formátum is, mellettük viszont a DB Nomics a JSON[16], általánosan használt formátumot támogatja.

Ennek használata számunkra is jelentősen leegyszerűsíti az adatelérést, amihez maga az oldal is biztosít Python könyvtárat[17].

A kliens kimenete a fent említett Python Pandas DataFrame lesz itt is.

2.3 CSV adatelérés

A CSV (comma separated values = vesszővel elválasztott adatok), vagy TSV (TAB separated values = tabulátorraal elválasztott adatok) egyszerű, szöveges adatformátumok, melyek valójában nem tekinthetőek teljes értékű adatbázisnak, viszont a platform és szoftverfüggetlen hordozhatóságuk okán a mai napig a legtöbb felület alapértelmezetten támogatja őket. Gyakorlatilag egy-egy adatbázistáblaként, vagy például egy Excel munkafüzetként tekinthetünk rájuk.

A megvalósított programba is bekerült az OECD exportálási funkciójával elérhető CSV formátum támogatása, ezzel közelítve a teljes offline működéshez is. Fontos megjegyezni, hogy a teljes offline működéshez szükséges lesz egy/több saját COCO modell megvalósítást is integrálni a programba, hisz a jelen megoldás a My-X COCO URL-en[18] keresztüli hívását használja fel az OAM táblák kiértékelésére.

[16] JSON (JavaScript Object Notation) (2022) https://www.json.org/json-en.html (Utolsó letöltés: 2022.11.15.)

[17] DBnomics Python Client (2022) https://git.nomics.world/dbnomics/dbnomics-python-client (Utolsó letöltés: 2022.11.15.)

[18] MyX-Free COCO (2022) https://miau.my-x.hu/myx-free/coco/ (Utolsó letöltés: 2022.11.15.)

3 A PROJEKT HÁTTERÉNEK BEMUTATÁSA

Miután felmértük a rendelkezésre álló adatvagyonok elérését a következő lépés volt felmérni, hogy milyen módon és mit is kell felépítenünk a projekt megvalósításához.

Fokozatosan, egyre mélyebben mértük fel és terveztük meg a probléma megoldásához szükséges feladatok lépéseit, ahogy a csapatunk felépítése is változott a kezdeti felálláshoz képest, ahogy egyre nagyobb érdeklődés mutatkozott a projekt által megvalósítandó feladat iránt.

3.1 Az automatizálandó feladat

Az előzményekben már többször előkerült a projekt magját adó alapötlet, a gazdasági adatok alapján a politikai mozgások becslése. Ennek különleges aktualitása a 2022 évi magyarországi parlamenti választások is voltak.

Ezzel kapcsolatban egy részletesebb dokumentum is a rendelkezésünkre állt dr Pitlik László és Pitlik Marcell jóvoltából[19]. Ebben a dokumentumban egy Excel táblázatban [20] került lépésről-lépésre levezetésre egy kiértékelés. Ezen táblázat prezentálása kapcsán merült fel több társammal közösen az automatizálás lehetősége.

Első lépésként lépésről-lépésre végig mentünk minden pontján a táblázatnak, egyrészt a megértés, másrészt a dokumentálás miatt is. Ezzel előállt a fő automatizálandó feladat:

- bemeneti adatok betöltése, releváns adatok kiválasztása a szűrési feltételek testre szabásával
- kimutatás tábla előállítása a bemenő adatokból
- a kimutatás tábla adatainak „tisztítása”
- OAM alaptáblázat előállítása a kimutatás tábla alapján

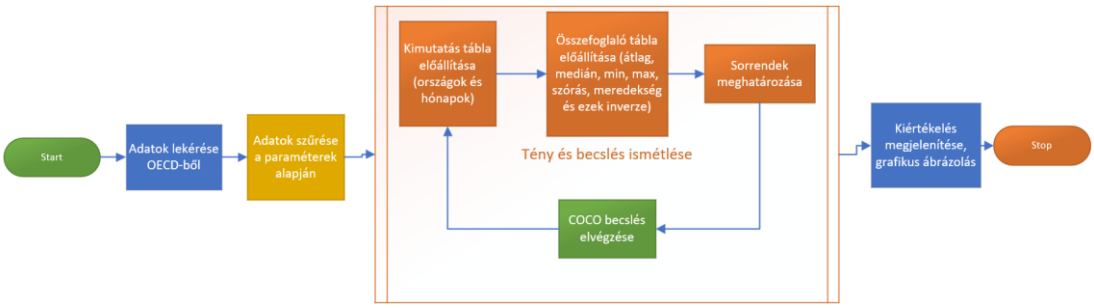
[19] Választás 2022 Magyarország – avagy Robotpolgár 2022 (Election 2022 Hungary or Robot-Citizen 2022) (2022) https://miau.my-x.hu/miau/281/election_2022_hu.docx (Utolsó letöltés: 2022.11.15.)

[20] https://miau.my-x.hu/miau/281/KEI_10012022150615288.xlsx (Utolsó letöltés: 2022.11.15.)

- az OAM adatok rangsorolt értékelése egy új táblázatban, különböző súlyozások alapján
- COCO alapú becslés az előállított OAM rangsorolt mátrix alapján[21]

Mivel ez egy adatsor kiértékelését jelentette, így ezen lépéseket kellett újra és újra ismételni, az általunk meghatározott mennyiségben.

Ezen lépések alapján állt össze a lépések folyamatábrája. [1. ábra - A program működésének folyamatábrája] Ez az ábra egy absztrakt képet fest a legfontosabb lépésekről, melyek a megvalósításhoz szükségesek.



1. ábra - A program működésének folyamatábrája (Forrás: saját munka)

A program indítása után az adatokat le kell kérni az OECD adatbázisából, majd kiszűrni a megadott paraméterek szerintieket. Ezután egy ismétlődő blokk következik, melyben a tény és becslésadatok meghatározása történik a kívánt mennyiségben. Amint az általunk elvárt mennyiségű adat a rendelkezésre áll, akkor pedig ebből előállítunk egy kiértékelést és ezt ábrázoljuk grafikusan is, majd a program befejezi a futását. Ez a vázszerkezet adja az automatizált megvalósítás végrehajtási folyamatát.

3.2 A csapat és a fejlesztői környezet felépítése

Előzőleg már említettem az agilis módszertanok használatát a projektünk kapcsán. Mivel a csapatunk sokféle háttérrel rendelkező, különböző szakmákból érkező hallgatótársakból állt, így ez elsősorban abban nyilvánult meg, hogy nem volt célunk

[21] A COCO módszer bemutatása; Pitlik Marcell – Térfogati égés optikai vizsgálata és mesterséges intelligencia alapú elemzése, 7. MELLÉKLETEK, pp 49-55. (2020) http://www.combustioninstitute.hu/tdk_dij/2021/PitlikMarcell_TDK2020.pdf (Utolsó letöltés: 2022.11.15.)

a csapaton belül hierarchiát kialakítani és mindenkiből a számára legoptimálisabb feladatkörrel felruházni. Az idő rövidsége miatt, például a Scrum[22] alapelveiből is igyekeztük például a rendszeres „ceremóniákat" átvenni, a folyamatos, nyílt egyeztetéseket, avagy a közös dokumentálást is. Nagy hangsúlyt fektettünk a csapatmunkára is, ezzel is motiválva és segítve egymást.

A mélyebb informatikai ismereteim és a technológiai háttér adottsága miatt én készítettem elő egy dedikált virtuális gépet a csapat számára. Mivel általánosságban véve is a nyílt forrású[23] szoftvereket részesítem előnyben, így egy Linux[24] disztribúció, konkrétan egy Ubuntu Server[25] operációs rendszer alatt valósult meg a környezet. Ezen immár a csapat egy-egy tagjával közösen alakítottuk ki a szoftverkörnyezetet.

Ez többek között egy GitLab[26] környezetet is jelentett, mely a git verziókezelésen[27] és a könnyen használható webes felületen kívül Kanban táblákat[28] és Wikipedia felületet és a háttérben egy PostgreSQL[29] adatbázist is biztosított számunkra. A Kanban tábla használatával tudtuk a részfeladatokat dokumentálni és azok állapotát nyomon követni.

Munkám során DevOps metódusok[30] szerint dolgozom, mely leegyszerűsítve annyit jelent, hogy egy (szoftver) termék teljes felügyelete a munka részét képezi. Azaz a környezet és a háttér biztosításán kívül a fejlesztés, a tesztelés és hibajavítás is egyaránt. Ezt teljes mértékig sajnos nem tudtuk implementálni a csapatunkban, de az egyes fázisok alapján legalább 1-1 csapattagot dedikáltunk az adott feladatra.

Így mivel például a szoftvertesztelés folyamata sem csak a statikus és az integrációs tesztekből áll, vagy akár a minőségi dokumentálás sem feltétlenül bárki

[22] What is Scrum? (2022) https://www.scrum.org/resources/what-is-scrum (Utolsó letöltés: 2022.11.15.)
[23] Open Source Initiative (2022) https://opensource.org/ (Utolsó letöltés: 2022.11.15.)
[24] Linux.org (2022) https://www.linux.org/ (Utolsó letöltés: 2022.11.15.)
[25] Ubuntu Server (2022) https://ubuntu.com/server (Utolsó letöltés: 2022.11.15.)
[26] GitLab (2022) https://about.gitlab.com/ (Utolsó letöltés: 2022.11.15.)
[27] git (2022) https://git-scm.com/about (Utolsó letöltés: 2022.11.15.)
[28] What is Kanban? Explained for Beginners. (2022) https://kanbanize.com/kanban-resources/getting-started/what-is-kanban (Utolsó letöltés: 2022.11.15.)
[29] PostgreSQL: The World's most advanced open source relational database (2022) https://www.postgresql.org/ (Utolsó letöltés: 2022.11.15.)
[30] What is DevOps? (2022) https://aws.amazon.com/devops/what-is-devops/ (Utolsó letöltés: 2022.11.15.)

számára könnyen megvalósítható dolog, így ezen feladatok ellátására is volt hallgatótársam.

3.3 Az adatok elérése és az abból előállt kimenet - OECD

Az OECD az adatok fejlesztői API-n keresztüli elérés esetén biztosít egy adott szűrés alapján előkészített hivatkozási webcím előállítást is. Ez a következő formátumban építi fel a címet:

https://stats.oecd.org/SDMX-JSON/data/KEI/<Indikátor(ok)>.<Országkód(ok)>.<Mérésjelölő(k)>.<Gyakoriság(ok)>/all?startTime=<Kezdődátum>&endTime=<Végdátum>&dimensionAtObservation=allDimensions

A lekérdezésben szereplő paraméterek a következők[31]:

- indikátor(ok): az elérhető gazdasági mutató(k) megadása (A „CL_KEI_SUBJECT” ID alatti szakasz)
- országkód(ok): az adatokat szolgáltató országok, vagy országcsoportok (például Eurózóna, vagy az EU 27 országa) (A „CL_KEI_LOCATION” ID alatti szakasz)
- mérésjelölő(k): a mérés az előző időszakokhoz viszonyított legyen-e, ez alapján lehet GP (Growth previous period – növekedés mértéke az előző periódushoz képest), GY (Growth on the same period of the previous year – növekedés mértéke az előző év azonos periódusához képest) és/vagy ST (Level, ratio or index – szint, arány vagy index) értékű (A „CL_KEI_MEASURE” ID alatti szakasz)
- gyakoriság(ok): éves (annual - A), negyedéves (quaterly - Q) vagy havi (monthly - M) szintű bontásban kérjük az idősoros adatokat (A „CL_KEI_FREQUENCY” ID alatti szakasz)

[31] Az adatstruktúra (2022) https://stats.oecd.org/restsdmx/sdmx.ashx/GetDataStructure/KEI (Utolsó letöltés: 2022.11.15.)

- kezdő és végdátum: lehet éves (pl 2022), lehet féléves (2022-S1 vagy 2022-S2), lehet negyedéves (2022-Q1 … Q4) és lehet havi (2022-01 … 12, vagy 2022-M1 … M12) formátumú a megadása

Egy paraméterből több érték megadása esetén „+” jel az elválasztás közöttük.

Lehetőségünk van tovább szűkíteni a kapott eredményeket a „dimensionAtObservation”, a „detail” és az „UpdatedAfter” megadásával[32].

Minden ilyen lekérdezésnek a kimeneti formátuma az SDMX formátum szabványos kimenete.

3.4 Az adatok elérése és az abból előállt kimenet – DB Nomics

A DB Nomics sok szolgáltató adatait gyűjti, egyszerű adatszinkronizálással, mely szolgáltatók közül az egyik az OECD és annak része a -jelenleg- számunkra fontos KEI adatvagyon is. Ezáltal az adatszerkezet felépítése csaknem megegyezik az OECD-nél elérhetővel.[33]

Ennek egy közvetlen, a DB Nomics verziókövető rendszerében elérhető URL-je: https://git.nomics.world/dbnomics-json-data/oecd-json-data/-/raw/master/KEI/dataset.json

A formázott JSON adatok közül egyszerűen kiolvasható a „dimensions_values_labels” szakasz alatt az OECD-nél is felsorolt „SUBJECT”, „LOCATION”, „FREQUENCY” és „MEASURE” is egyaránt.

Ezen adatok segítségével az OECD URL szerkezetéhez hasonlóan, viszont jobban strukturált módon építhető fel az API hívás URL-je:

https://api.db.nomics.world/v22/series/OECD/KEI?dimensions={"SUBJECT":["indikátor1","indikátor2"],"LOCATION":["országkód1","országkód2"],"MEASURE":["mérésjelölő1","mérésjelölő2"],"FREQUENCY":["gyakoriság1","gyakoriság2"]}&observations=1

[32] Az OECD SDMX JSON API ismertetője (2022) https://data.oecd.org/api/sdmx-json-documentation/ (Utolsó letöltés: 2022.11.15.)

[33] A DB Nomics adatstruktúra (2022) https://api.db.nomics.world/v22/datasets/OECD/KEI (Utolsó letöltés: 2022.11.15.)

Ezután a „dimensions” változó egyértelmű értékadásához az URLEncode megoldást kell alkalmazni[34], mely a fenti URL alapján ezt adja eredményül:

https://api.db.nomics.world/v22/series/OECD/KEI?dimensions=%7B%22SUBJECT%22%3A%5B%22indik%C3%A1tor1%22%2C%22indik%C3%A1tor2%22%5D%2C%22LOCATION%22%3A%5B%22orsz%C3%A1gk%C3%B3d1%22%2C%22orsz%C3%A1gk%C3%B3d2%22%5D%2C%22MEASURE%22%3A%5B%22m%C3%A9r%C3%A9sjel%C3%B6l%C5%911%22%2C%22m%C3%A9r%C3%A9sjel%C3%B6l%C5%912%22%5D%2C%22FREQUENCY%22%3A%5B%22gyakoris%C3%A1g1%22%2C%22gyakoris%C3%A1g2%22%5D%7D&observations=1

Az URLEncode mindössze a paraméterül kapott ASCII [35] értékeket HTML karakterkódokká konvertálja, hogy a paraméterekben szereplő esetleges speciális karakterek ne kerüljenek feldolgozásra a szerver oldali kiértékelés esetén. Más, hasonló és elterjedt módszer a Base64Encode[36], mikor Base64 kódolást és a szerver oldalon dekódolást végzünk a paraméterek értékein. Mindkét technikát aktívan használják hackerek is, amennyiben egy webes HTML Form, vagy API nincs megfelelően védve.

A lekérdezés kimenete pedig egy standard JSON lesz.

Hogy ne kelljen ezt a kimenti adathalmazt értelmezni és feldolgozni is, így a DB Nomics elérhetővé tette [37] a saját kliensét Python programozási nyelvhez, mely segítségével a fenti API hívás leegyszerűsödik és a kimentünk egy Pandas DataFrame objektum lesz a JSON helyett.

[34] URLEncode ismertető – w3schools (2022) https://www.w3schools.com/tags/ref_urlencode.ASP (Utolsó letöltés: 2022.11.15.)

[35] ASCII tábla (2022) https://www.ascii-code.com/ (Utolsó letöltés: 2022.11.15.)

[36] Base64 leírás a Mozilla oldalán (2022) https://developer.mozilla.org/en-US/docs/Glossary/Base64 (Utolsó letöltés: 2022.11.15.)

[37] DB Nomics adatletöltési segédlet (2022) https://db.nomics.world/docs/download-data/ (Utolsó letöltés: 2022.11.15.)

4 A PROGRAM FELÉPÍTÉSE

„Tudás csak az, ami forráskódba átírható, minden más emberi aktivitás, művészet.”

Donald Ervin Knuth

Knuth szavai alapján indult el az általunk elemzett probléma kisebb egységekre bontása, míg kellően algoritmizálhatóvá nem vált. Így lehet a komplex problémákat feldolgozni és a gondolataink sokaságát leképezni algoritmusokká.

A következő alfejezetekben a tényleges programmegvalósítás bemutatása következik, fókuszálva a speciális megoldásokra és a felépítés elemeire.

4.1 A program fájl- és könyvtárstruktúrája

A program elkészítése során igyekeztem a modularitást szem előtt tartani, hogy az egyes komponensek más megoldásokban is könnyen felhasználhatók legyenek, avagy könnyen bővíthető/módosítható/lecserélhető legyen bármelyik, amennyiben egy hatékonyabb, vagy számunkra optimálisabb megoldást sikerül találnunk.

Az osztályok használata nem adott volna többletértéket a jelen megoldásunkhoz, így az osztályokba szervezést a probléma túlkomplikálásának értékeltem, emiatt egyszerűen függvények hívásával valósul meg a különböző fájlok összekapcsolása. Ez önmagában nem okoz semmilyen hátrányt, sőt adott esetben még kisebb erőforrásfoglalással is járhat.

A program a főprogramból (e_gazsag_lite.py) és a modules könyvtár alatt elhelyezkedő fájlokból áll.

A fájlok teljes listája a következő:

```
e_gazsag_lite.py
kei_label_data.json
modules/__init__.py
modules/calculate.py
modules/collect_input.py
```

```
modules/input_csv.py
modules/input_dbnomics.py
modules/input_oecd.py
modules/prepare_data.py
```

A főprogram a „modules/calculate.py” meghívására épül, mely a begyűjtött adatok tényleges kiértékelését végzi.

A „modules/calculate.py” meghívja a modules/prepare_data.py fájlt, így készítem elő az OAM táblázatot, majd ugyanezen fájlban hívom meg és értékelem ki a COCO becsléseket is.

A modules/prepare_data.py hívja meg a modules/collect_input.py fájlon keresztül a paraméterként beadott CSV, OECD vagy DB Nomics bemenetet (modules/input_*.py), illetve készíti elő az elérhető program paramétereket is.

A program paraméterek egy helyben tárolt JSON fájlba kerülnek kiírásra (kei_label_data.json), mely ilyen szempontból kakukktojásnak számít, hisz nem programkód és a tartalma változhat, amennyiben bővül az adatvagyon például további indikátorokkal.

A futás során a program a paraméterezéstől függően további fájlokat állíthat elő, mely a kiértékelés eredményét rögzíti CSV és PNG formátumban.

4.2 A program működése részletesen

A kiinduló Excel táblázat alapján feltérképezett egymás utáni lépések megvalósítása több irányból indult el. Először is fel kellett mérni, hogy milyen módon tudjuk az adatokat az adatbázisokból kinyerni, majd ezt követően fontos volt felmérni, hogy milyen paraméterezésre van szükség a feladatok elvégzéséhez.

Ez természetesen csak a kiindulási alap, mely mögé fel kellett építeni egy hagyma héjszerkezetéhez hasonlóan a többi részegységet, melyek bemutatása következik a következő alfejezetekben.

4.2.1 A főprogram paraméterezése

A paraméterek tekintetében az OECD által biztosított szűrési feltételek mellett szükséges volt a bemeneti formátum azonosítása, mely a CSV esetén a bementi fájl nevét és az elválasztó karaktert is további paraméterként megadhatóvá kell tegye.

Míg például, amennyiben valaki nem néz után az OECD bemenő paramétereinek, akkor azokat is a felhasználó rendelkezésére kell bocsátani.

Ezek alapján a „--help”, vagy „-h” paraméter használatával a következő paraméterezési opciók érhetőek el:

--input-source, -i: a bemeneti adatok forrása, lehet OECD, DBNomics és CSV

--csv: amennyiben a bemeneti formátum CSV, akkor a CSV fájlnév

--csv-delimiter: a CSV file elválasztó karaktere, alapértelmezetten TAB

--subject, -s: az indikátor, amit vizsgáljunk

--location -l: az országkód, melyre készüljön a becslés és tény összehasonlítás

--measure, -m: a mérőkód a szűréshez, értéke az OECD szerinti GP, GY, ST lehet

--frequency, -f: a gyakoriság a szűréshez, értéke az OECD szerinti A, Q, M lehet

--start-date, -sd: a kezdődátum a kiértékelés alapjául szolgáló idősorhoz

--end-date, -ed: a végdátum a kiértékelés alapjául szolgáló idősorhoz

--output-count, -n: hány becslést hasonlítsunk össze a tény adatokkal? alapértelmezetten 10

--output-prefix: egyedi előtag a kimeneti fájlokhoz

--output-suffix: egyedi utótag a kimeneti fájlokhoz

--save-output: mentsük-e a kimeneti fájlokat, True/False, alapértelmezetten True

--debug: debug mód használata, True/False, alapértelmezetten False

--print-labels: az OECD elérhető opcióit listázza, True/False, alapértelmezetten False

Természetesen ezen paramétereket lehetne tovább bővíteni, akár több érték megadhatóságával is, de ez nem volt elsődleges célja a projektnek jelen állapotában.

A főprogram ismertetésére ezen fejezet végén visszatérek, miután a program logikáját részletesen bemutattam.

4.2.2 Az elérhető paraméterek lekérése

A „--print-labels True” kapcsoló mögötti logika megvalósítása következett, mely a DB Nomics felületéről tölti le egy JSON fájlba az elérhető indikátorokat, országjelölőket, mérésjelölőket és gyakoriságokat is.

A „read_labels” függvényből származó, következő kódrészlet vizsgálja meg, hogy létezik-e a fájl, ha létezik, 0 bájt méretű-e, illetve a létrehozás időpontja 1 napnál (60*60*24=86400 másodpercnél) régebbi-e?

```
if (not os.path.exists('kei_label_data.json')) or
(os.stat('kei_label_data.json').st_size==0 or
(os.stat('kei_label_data.json').st_mtime < time.time() -
86400)):
```

Amennyiben ezen feltételek közül bármelyik is teljesül, akkor a program megpróbálja letölteni és eltárolni a „kei_label_data.json” fájlt, majd pedig feldolgozni azt. Amennyiben a feltételek nem teljesülnek, akkor közvetlenül a feldolgozás következik. Ez például annyi rizikót hordoz magában, hogy amennyiben akár csak 1 bájt adat is letöltésre került, viszont a fájl tartalma nem szabványos JSON, akkor a program hibát észlelve kiléphet.

A feldolgozást követően pedig visszatér a függvény a beolvasott adatokkal, külön dictionary változókban.

Ezen változókat tölti be és jeleníti meg a „print_labels” függvény.

4.2.3 A forrás kiválasztása és a paraméterek átadása

Mint már írtam, egy absztrakt osztály jelleggel valósítottam meg a különböző források kezelését. Mindez a „modules/collect_input.py” fájlban található „get_input_data” függvényben kerül megvalósításra. A paraméterek a szűrési feltételeken kívül a CSV, OECD, vagy DBNOMICS „input_source” meghatározását és a CSV bemenet esetén a CSV fájl helyét és nevét, opcionálisan a CSV elválasztó karaktert tartalmazzák.

A megfelelő bemenet kiválasztása után azon függvények visszatérési értéke megegyező lesz, így a kimenet egy Pandas DataFrame objektum, az OECD fejlécei alapján a "SUBJECT", a "LOCATION", a "MEASURE", a "FREQUENCY", a "TIME_PERIOD" és a "value" oszlopokkal.

4.2.4 Az adatok betöltése CSV-ből

Python-ban a CSV fájlok kezelése nagyon egyszerűen megvalósított, elérhető hozzá a „csv”[38] nevű könyvtár, mely segítségével lehetőségünk van a csv.reader[39] függvény használatára, ami kezel az elválasztó karakteren túl számos más CSV formázási megvalósítást is.[40]

Mivel számunkra elegendő volt az OECD formátum támogatása, így az elválasztó karakter megadása elégséges volt a paraméterezésnél.

A „modules/input_csv.py” fájlban a „get_input_from_csv” függvény paraméterezése után töltöm be a CSV értékeit. Először ellenőrzésre kerülnek az oszlopfejlécek a fájlban, majd az adatokat a DB Nomics-al egységes formátumra formázom, betöltöm egy list objektumba és szűrés után egy Pandas DataFrame objektummal tér vissza a függvény az előzőleg ismertetett (4.2.3) oszlopokkal.

4.2.5 Az adatok kinyerése API híváson keresztül – OECD

Az OECD adatok eléréshez („modules/input_oecd.py”) a PandaSDMX könyvtárt használtam fel, mely elméletben egy egyszerű és gyors hozzáférést biztosít az adatvagyonhoz. Mivel az összes KEI adatsor közel kétmilliós tétel, így ezen adatok összeválogatása és rendelkezésre bocsátása erőforrásigényes feladat. Ez több problémát is okozott, pláne a nagyobb időintervallumok lekérdezése esetén.

Egyrészt önmagában az adatok lekérése is időnként több percig tartott, másrészt számos alkalommal szakadt meg időközben tisztázatlan okból az adatkapcsolat.

Ezt teljes mértékig kivédeni sajnos nem lehet. Mivel az adatok HTTP protokollon, azaz TCP-n keresztül kerülnek átadásra, így Python-ban mód nyílik a TCP socket kapcsolatának a mélyebb beállítására. Ehhez a „socket”[41] és az „urrllib3”[42] nevű

[38] CSV könyvtár (2022) https://docs.python.org/3/library/csv.html (Utolsó letöltés: 2022.11.15.)
[39] CSV Reader függvény (2022) https://docs.python.org/3/library/csv.html#csv.reader (Utolsó letöltés: 2022.11.15.)
[40] CSV támogatott formázási lehetőségek (2022) https://docs.python.org/3/library/csv.html#csv-fmt-params (Utolsó letöltés: 2022.11.15.)
[41] Python socket module (2022) https://docs.python.org/3/library/socket.html (Utolsó letöltés: 2022.11.15.)
[42] URLLib3 (2022) https://urllib3.readthedocs.io/en/stable/ (Utolsó letölés: 2022.11.15.)

könyvtárakra volt szükség, utóbbiból csak a „Connection” alá tartozó „HTTPConnection”[43] készletre.

Ezzel lehetőségem nyílt a következő kódrészlettel megnövelni a socket kapcsolat nyitvatartását:

```
HTTPConnection.default_socket_options = (
    HTTPConnection.default_socket_options + [
        (socket.SOL_SOCKET, socket.SO_KEEPALIVE, 1),
        (socket.SOL_TCP, socket.TCP_KEEPIDLE, 120),
        (socket.SOL_TCP, socket.TCP_KEEPINTVL, 30),
        (socket.SOL_TCP, socket.TCP_KEEPCNT, 10)
    ]
)
```

Ez önmagában sajnos nem oldott meg minden problémát, így a tényleges adatlekérés során is tovább kellett finomítani a lekérdezést. Így a PandaSDMX kapcsolat felépítésekor beállítottam SQLite[44] adatformátumú gyorsítótárt és 10 perces időtúllépést is a kapcsolathoz a következő módon:

```
sdmx.Request('OECD',       timeout=600,       backend='sqlite',
fast_save=True,       expire_after=3600,       use_cache=True,
toFile="OECD_SDMX")
```

Ezen megoldások segítségével sikerült alapvetően stabilabbá tenni a közvetlen OECD lekérdezéseket. Ennek ellenére a lekérdezés sebessége a gyorsítótár használata ellenére sem túl ideális.

A PandaSDMX lekérdezés után egy DataFrame objektumot kapunk eredményül, mely a bemeneti paraméterezés alapján szűrésre kerül és így adjuk vissza kimenetnek. Ezzel az elvileg legegyszerűbb lekérdezési mód jelentősen komplexebben ugyan, de működőképes bemenet lett.

[43] UELLib3 HTTPConnection (2022) https://urllib3.readthedocs.io/en/stable/reference/urllib3.connection.html#urllib3.connection.HTTPConnection (Utolsó letöltés: 2022.11.15.)

[44] SQLite (2022) https://www.sqlite.org/index.html (Utolsó letöltés: 2022.11.15.)

4.2.6 Az adatok kinyerése API híváson keresztül – DB Nomics

Mint már előzőleg is írtam, mind az OECD, mind a DB Nomics biztosít API elérést az adatvagyonukhoz. A saját tesztelés tapasztalatai alapján az OECD adatkapcsolata lassabb és egyben instabilabb is volt.

Épp emiatt merült fel szinte a kezdetektől az alternatív lehetőség a DB Nomics használata. Természetesen a szinkronizált adatok egyben azt is jelentik, hogy a megvalósítás milyensége esetlegesen ronthat az adatvagyon értékén, például kisebb pontosságú tizedesértékeket találunk, vagy esetlegesen egy kerekítési algoritmus módosít az adatokon.

A DB Nomics épp ezt próbálja azzal kivédeni, hogy szabadon elérhetővé tette nyílt forrású alapon[45] az adatletöltő programjait, így az „OECD fetcher”[46] programját is.

Így immár nyugodtan felhasználhattuk bemeneti adatvagyonnak ezt a forrást is, mely már az első próbálkozások esetén is nagyságrendekkel gyorsabb lekéréseket biztosított.

További könnyebbség volt, hogy hivatalos Python könyvtárral[47] támogatják az adatok lekérését, így az OECD KEI adatok elérése ez a parancs:

```
dbnomics.fetch_series("OECD",    "KEI",    series_code=subject,
max_nb_series=1000000)
```

Itt a „series_code” paraméter segítségével máris szűröm a bemeneti „SUBJECT” értéke alapján a KEI adatokat, ezáltal csökkentve a feldolgozás és a letöltés idejét.

A kliens nagyságrendekkel stabilabban és gyorsabban is működött az OECD PandaSDMX lekérdezésétől és a „fetch_series” kimenete ennek is egy DataFrame objektum, így egyszerű volt a számunkra értékes oszlopokat kiszűrni, mely oszlopokból az egységesítés érdekében az „original_value” fejlécet átneveztem „TIME_PERIOD”-ra és visszaadtam a paraméterek alapján leszűrt DataFrame objektumot.

[45] DB nomics GitLab (2022) https://git.nomics.world/dbnomics (Utolsó letöltés: 2022.11.15.)

[46] OECD fetcher (2022) https://git.nomics.world/dbnomics-fetchers/oecd-fetcher/ (Utolsó letöltés: 2022.11.15.)

[47] dbnomics-python-client (2022) https://git.nomics.world/dbnomics/dbnomics-python-client (Utolsó letöltés: 2022.11.15.)

4.2.7 Az OAM előkészítése

A program legösszetettebb része az előzőleg Excel-ben megvalósított OAM felépítése a bemeneti adatok alapján. Az előzőekben bemutatott adathalmaz felépítése nem foglalkozott még az adatok minőségével vagy a folytonosságukkal.

Így, miután a bementi adatok betöltését egy kimutatási táblává változtatom (vö Pivot Table[48]), már ennek a létrehozása során a „Nem Elérhető” (Not Available, „na”) értékeket eldobom:

```
res = input_df.pivot_table(index=["LOCATION"], columns=['TIME_PERIOD'], values=["value"], fill_value=None, aggfunc=sum, dropna=True)
df = np.round(res.dropna(), 1)
```

Az Excel alapján felmért célérték meghatározás (Y) a dátum alapján az utolsó teljes értékű oszlop, így ezt kiemelem a kimutatási táblázatból az „oam_result” változóba, majd előkészítem a maradék táblázat adataiban az Excel szerinti átlag, medián, minimum, maximum, variancia, standard variancia és meredekség értékeket:

```
# Átlag
oam_mean = df.mean(axis=1).to_dict()
# Medián
oam_median = df.median(axis=1).to_dict()
# Minimum
oam_min = df.min(axis=1).to_dict()
# Maximum
oam_max = df.max(axis=1).to_dict()
# Variancia
oam_var = df.var(axis=1).to_dict()
# Standard eloszlás
oam_std = df.std(axis=1).to_dict()

# Slope, avagy meredekség
xtomb=[]
for elem in range(1,len(df.columns)+1):
    xtomb.append(elem)
```

[48] Pivot Table (2022) https://support.microsoft.com/en-us/office/create-a-pivottable-to-analyze-worksheet-data-a9a84538-bfe9-40a9-a8e9-f99134456576 (Utolsó letöltés: 2022.11.15.)

```
oam_slope = {}
for index, row in df.iterrows():
    ytomb=row['value']
    oam_slope[index]=(round(np.polyfit(xtomb, ytomb, 1)[0], 2))
del (xtomb)
```

Mivel az eredményeket egy táblázatban szerettem volna elérni, így dictionary típusú változóba tároltam el a műveletek eredményeit. Ez abban segített, hogy a hivatkozási értékek minden változóban megegyezőek voltak a kiindulási Pivot táblázatban szereplőkkel. Így ezek összesítésével sikerült felépítenem az OAM táblázatot:

```
# OAM tábla előállítása
oam_table = {}
oam_y = {}
for key in oam_result.keys():
    oam_table[key]=[oam_mean[key], oam_median[key], oam_std[key], oam_max[key], oam_min[key], oam_var[key], oam_slope[key], round(oam_result[key]*1000+1000000)]
    oam_y[key]=round(oam_result[key]*1000+1000000)

# Alakítsuk át DataFrame-re
oam_table_df = df.from_dict(oam_table, orient='index')
oam_table_df.columns = ["mean", "median", "std", "max", "min", "var", "slope", "Y"]
```

Itt már, az Y értékét a 0-nál nagyobb egész számmá alakítom át, ezért szorozzuk meg 1000-el, majd adunk hozzá 1000000-t.

A következő lépés az objektumok attribútumok szerinti sorszámozása, azaz az egyes országok rangsora a különböző számítások szerinti kiértékelésben az OAM táblázat alapján.

Erre a legegyszerűbb megoldást a DataFrame „rank” metódusa adta, melynél lehetőségünk van a sorszámozás irányát és a módját is megadni. A mód meghatározásánál az „average”, azaz átlag használata eredményezte a kiindulási Excel-ben szereplő adatokkal az egyezést, így ezt használtam fel. A sorszámozás iránya pedig rögtön a kezembe adta az ellenőrzésre szolgáló inverz sorszám értékeket is.

Ezen adatok felhasználásával több táblázatot építek fel és adok vissza kimenetként a további feldolgozásra.

4.2.8 A COCO kiértékelés URL alapú meghívása

A COCO módszert[49] (Component-based Object Comparison for Objectivity) Pitlik László fejlesztette ki, mely egy lineáris programozási algoritmus alapú hasonlóságelemzési módszer. A módszer rövid bemutatása Pitlik Marcell jóvoltából magyarul is elérhető.[50]

Erre a módszerre épülő algoritmusokat ingyenesen, online elérhetővé tettek a My-X oldalain, a https://miau.my-x.hu/myx-free/coco/ URL alatt. A dolgozatomhoz a COCO STD került felhasználásra, távoli híváson keresztül a „modules/prepare_data.py" fájlban szerepló „coco_url" függvény segítségével.

Ehhez először az előzőekben előállított OAM táblázatot minimálisan át kellett formázni, mégpedig egy TAB karakterrel elválasztott és Windows-os sorvége jelekkel (CR+LF, azaz \r\n) ellátott táblázatra.

Ez egy egyszerű dupla „for" ciklussal került megvalósításra:

```
matrixstr=""
for row in matrix:
      cols = len(row)
      colcount = 0
      for column in row:
            matrixstr=matrixstr+str(column)
            colcount += 1
            if colcount < cols:
                  matrixstr=matrixstr+'\t'
      matrixstr+='\r\n'
```

[49] Pitlik László: Automated Generating of problem-specific Function for Forecasting and Decision Making orig.: Automatisierte Generierung problemspezifischer Prognosefunktionen zur Entscheidungsunterstützung, Wissenschaftlicher Fachverlag, Giessen, Germany ISBN 3-928563-60-2, 1993.

[50] A COCO módszer bemutatása; Pitlik Marcell – Térfogati égés optikai vizsgálata és mesterséges intelligencia alapú elemzése, 7. MELLÉKLETEK, pp 49-55. (2020) http://www.combustioninstitute.hu/tdk_dij/2021/PitlikMarcell_TDK2020.pdf (Utolsó letöltés: 2022.11.15.)

Ezután a COCO URL alapú meghívásához még további két paraméterre volt szükségem, amikkel meghatároztam a használt modellt és a webes form elküldését szimuláltam (HTTP GET Request-et küldtem):

```
data = {
            'modell': 'STD',
            'button2': 'Futtatás',
            'matrix': matrixstr
            }
```

A meghívott URL alapértelmezetten a https://miau.my-x.hu/myx-free/coco/engine3_curl_stairs.php, mely a lépcsőket is visszaadja. (Erre a továbbfejlesztésnél még szükségünk lesz, ellenőrzések futtatása miatt.)

A Python Pandas könyvtára képes közvetlenül HTML beolvasására is, ami nagyon hasznos volt számomra az eredmény kiértékelése szempontjából, mivel így a visszaadott táblázatokra közvetlenül tudtam hivatkozni és csak a számomra fontos, negyediket egyszerűen ki tudtam szűrni. Ebből a kinyert becslési és delta értéket adom vissza a függvénynek.

4.2.9 A különálló függvények egyesítése

Utoljára hagytam, miközben ez a „legkülső” függvény a program logikájában, mely az előzőekben bemutatott függvényeket hívja meg paraméterezve.

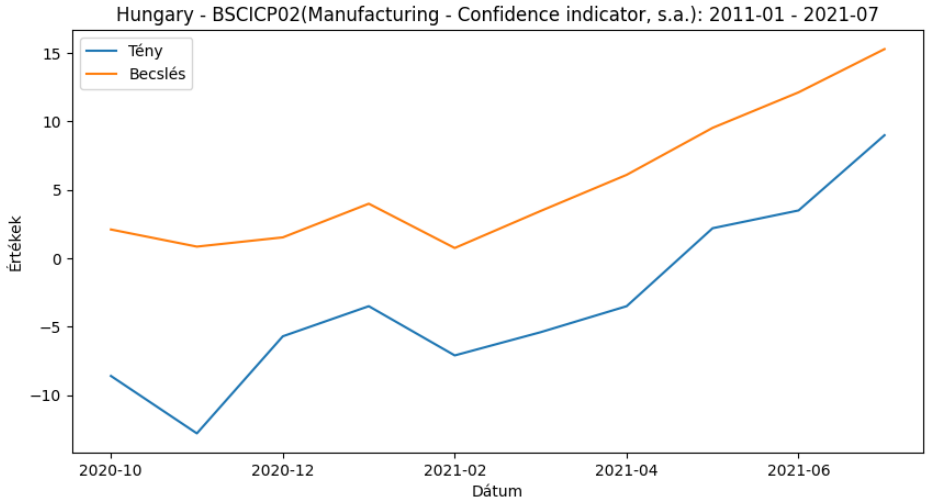
Ez a „modules/calculate.py” „calculate_estimation” függvénye, mely valóban csak a többi függvény meghívását végzi, majd visszatér a tény, a becslés és a delta értékekkel dictionary változók formájában.

4.2.10 A főprogram futása, a prezentáció

A 4.2.9-ben bemutatott becslés kalkulációt indító függvény meghívása az „e_gazsag_lite.py” „main” függvényén keresztül valósul meg. Ebben hívom meg a kiértékeléseket a paraméterezés szerinti darabszámban és időintervallumban.

A „modules/calculate.py” „calculate_estimation” függvény kimenetét eltárolom list változóba, melyből később DataFrame-et állítok elő az egyszerűbb eltárolás miatt.

Rendezem fordított irányba dátum szerint, majd előkészítem a DataFrame-et a megjelenítéshez és a MatPlotLib könyvtár segítségével megjelenítem a becslés-tény értékeket egy vonaldiagramon. [2. ábra]



2. ábra – A program által generált vonaldiagram (Forrás: saját munka, OECD adatok alapján)

4.3 A kódminőség javítása – statikus tesztelés

Egy program kapcsán rendszeresen felmerül, hogy a kódját egy fejlesztés során többször olvassák el, mint amennyit írják, így egyrészt a jól tagolt felépítés, másrészt a beszédes változónevek használata is fontos.

Ennek okán egy időben minden egyes cég, vagy akár fejlesztő is megalkotta a „saját” kódolási stílus előírásait. Ezek között jelentős átfedések mutatkoztak és például Python esetében a behúzás megengedő volta miatt előfordulhatott, hogy több fejlesztő ugyanazon kódbázisban különböző behúzásokkal dolgozzon, például valaki TAB-ot használt, míg más 4 db szóköz billentyűt.

Ez egy komplex kódbázisban jelentős hátrányt okozhatott, emiatt a Python megalkotója Guido van Rossum[51] és számos más fejlesztő szabványosítani kezdte[52] a követelményeket, így jött létre többek közözz a PEP8[53] ajánlás/szabvány is.

A Python fejlesztőközösség nemcsak használni kezdte el az ajánlásokat, hanem különböző eszközökkel támogatni is.

A fejlesztés során én is alkalmaztam a „black”[54] nevű eszközt, mely segít az egységes kód megjelenítésében. A használata kifejezetten egyszerű, csak egy cél fájlt, vagy egy cél könyvtárt kell megadni és automatán elvégzi a szabványosítást. Ez persze például a változónevekre nem fog kiterjedni, de az egységes megjelenést már garantálni fogja. Így egy komplexebb kód ellenőrzés első lépésének akár automatizáltan is bevethető eszközről van szó.

A „black” nem végez ellenőrzéseket, arra egy más eszközt célszerű bevetni, mely elérhető eszközök közül talán „pylint”[55] a legelterjedtebb. Ez az eszköz kifejezetten a fent említett PEP8 szerint ellenőrzi az előállított kódunkat és a felmerült hibákat, ajánlásokat listázza nekünk, hogy javíthassuk azokat. Ha minden esetben nem is érjük el a 10.0-át, ami a megkapható legmagasabb értékelés, akkor is ajánlott egy-egy fejlesztőcsapatnál meghúzni egy minimum határt a kódminőség biztosítása érdekében.

A „pylint” jelöli a minőségváltozásokat is az előző állapothoz képest, így a hibák javítása által láthatjuk, hogy mennyit javult a kódunk. Egy példa a futására, a „modules/input_oecd.py” fájlon:

```
> pylint.exe .\modules\input_oecd.py
************* Module modules.input_oecd
modules\input_oecd.py:15:0: R0913: Too many arguments (6/5)
(too-many-arguments)
modules\input_oecd.py:35:8: W0621: Redefining name 'logger'
from outer scope (line 160) (redefined-outer-name)
```

[51] Guido van Rossum – Personal Home Page (2022) https://gvanrossum.github.io// (Utolsó letöltés: 2022.11.15.)

[52] Python Enhancement Proposals (2022) https://peps.python.org/pep-0000/#introduction (Utolsó letöltés: 2022.11.15.)

[53] PEP8 – Style Guide for Python Code (2001.07.05.) https://peps.python.org/pep-0008/ (Utolsó letöltés: 2022.11.15.)

[54] Black – The Uncompromising Code Formatter (2022) https://github.com/psf/black (Utolsó letöltés: 2022.11.15.)

[55] Pylint – Star your Python code! (2022) https://www.pylint.org/ (Utolsó letöltés: 2022.11.15.)

```
modules\input_oecd.py:72:0: R0913: Too many arguments (6/5)
(too-many-arguments)
modules\input_oecd.py:108:8: W0621: Redefining name
'start_time' from outer scope (line 167) (redefined-outer-name)
modules\input_oecd.py:111:8: W0621: Redefining name 'logger'
from outer scope (line 160) (redefined-outer-name)


------------------------------------------------------------------
----
Your code has been rated at 9.12/10 (previous run: 9.12/10,
+0.00)
```

A fenti hibákat például ignoráltam, mivel a változók újra definiálása és a paraméterek mennyisége is szándékos volt a kódban. Természetesen, lehetne ezen is változtatni, de önmagában nem okoz minőségvesztést, ha akár az automatikus ignorálási listába teszünk bizonyos ellenőrzési feltételeket.

4.4 Dokumentáció generálása

Szinte minden fejlesztő számára az egyik legmegterhelőbb dolog a dokumentáció elkészítése. Az sem jó, mikor minden egyes sort megjegyzésekkel lát el valaki, vagy például egy függvény neve nem elég beszédes, esetleg megtévesztő is.

Ennek kapcsán létezik Python alatt a PEP257-ben tárgyalt DocString[56] fogalma. Önmagában nem jelent többet, mint speciális megjegyzések hozzáfűzését a függvényeinkhez és a fájljainkhoz.

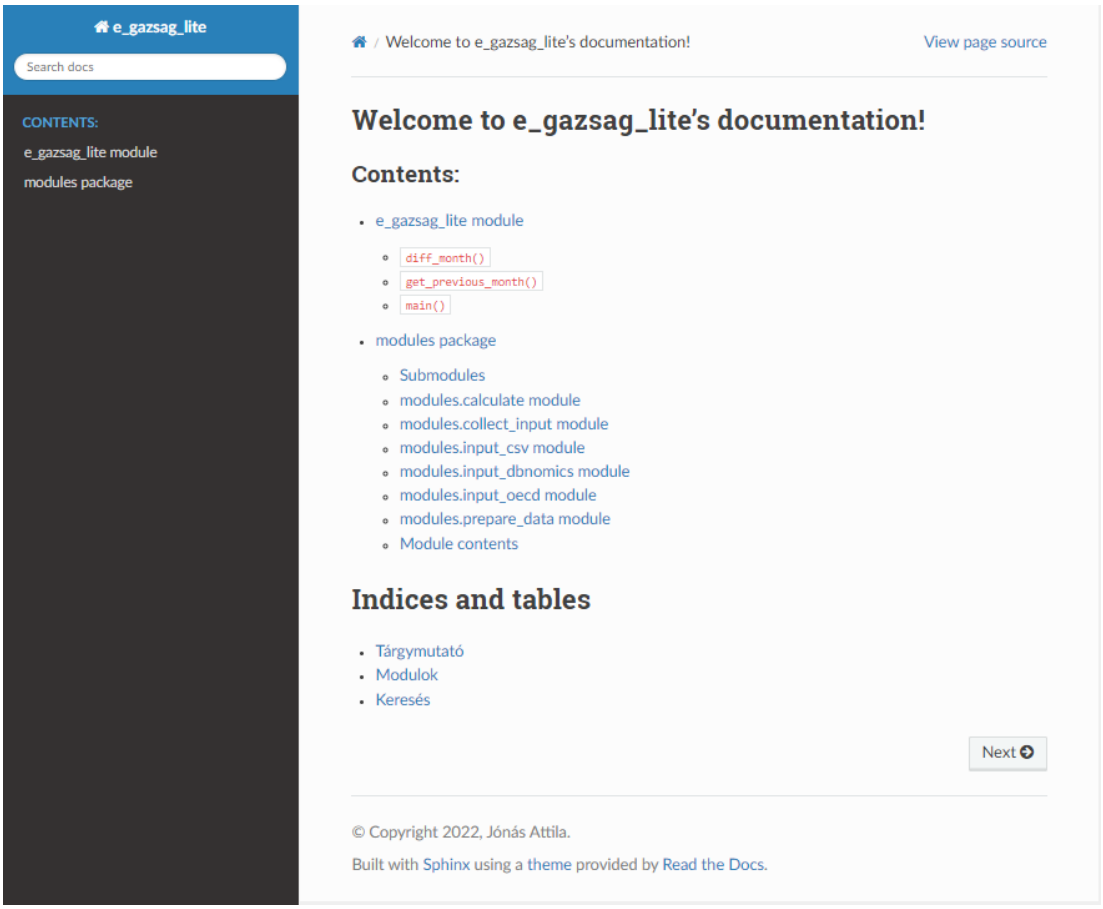
Amennyiben viszont ennek a használatával egészítjük ki a kódunkat (amit „pylint” is ellenőriz például), akkor egyrészt hónapokkal, vagy évekkel később megnézett régi kódban is rögtön egyértelmű lesz minden paraméter és hogy mit is csinál egy adott függvény, másrészt, például a fejlesztői környezetek legtöbbje támogatja a DocString használatát és egy függvényhívásnál megmutatja a leírását, értelmezi a paramétereket is. Még a legegyszerűbb függvények kapcsán is hasznos:

```
def diff_month(date1: date, date2: date) -> int:
    """
    Calculate difference between months
```

[56] PEP257 – Docstring Conventions (2001.05.29) https://peps.python.org/pep-0257/ (Utolsó letöltés: 2022.11.15.)

```
    :param date1:First date, to be reduced
    :param date2:Second date, to be subtracted
    :output:Integer value with the result of the subtraction
    """
    return (date1.year - date2.year) * 12 + date1.month -
date2.month
```

Egy további, jelentős előny még, hogy több eszköz is rendelkezésre áll a DocString-ek felhasználásával egy automatizált dokumentáció elkészítésére. Ezek közül én a „Sphinx”-el[57] generáltam a program dokumentációját egy statikus HTML formájába.[3. ábra]



3. ábra – e_gazsag_lite dokumentáció főoldala (Forrás: saját munka)

[57] Sphinx – Python Documentation Generator (2022) https://www.sphinx-doc.org/en/master/ (Utolsó letöltés: 2022.11.15.)

Például egy ilyen eszköz segítségével könnyen nyílik módunk egy átfogó fejlesztői dokumentáció elkészítésére, mely például csapatban együttműködve mindenképp hasznos és fontos dolog.

5 TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A fentiekben megvalósított program jóval túlmutat egy Proof of Concept megoldáson, azaz nem egyszerűen egy automatizáció működőképességét mutatja meg, hanem már tényleges kiértékelésekre is képes. Ennek ellenére messze nem teljes, vagy befejezett. Ahogy szinte megszámlálhatatlan további területen lenne alkalmazható egy hasonló. Avagy erre épülő megoldás is, a teljesség igénye nélkül a MIAÚ oldalain elérhető https://miau.my-x.hu/miau2009/index_tki.php3 listában számos OAM alapokon nyugvó, becslési probléma is szerepel.

Következzen néhány továbbfejlesztési lehetőség, mely lista még (jelentősen) bővülhet:

- A manuálisan elvégzett statikus tesztelések (4.3) után az unittesztelés[58] bevezetése is fontos lenne a program kódbázisszempontjából, amihez tesztesetek definiálására is szükség van. Ezzel jelentősen lehetne javítani a kód minőségét, hibakeresést is lehetne végezni és a későbbi fejlesztések nem törnék meg a program működését. (vö TDD – Test Driven Development[59])
- Hallgatótársaimmal továbbra is együtt dolgozva szeretnénk finomítani az indikátorok használatát és meghatározni egy-egy részautomatizált kimeneti automata szöveges értékelést (vö. hermeneutikai modul) is hozzájuk. Ehhez immár szükséges lenne az indikátorok helyben tárolása, hozzájuk például a tény-becslés értékpárok pozitív, vagy negatív irányú eltérésének a „jóságát” is tárolni kellene.
- Továbbá az OECD számos más, jelentősen szélesebb körű gazdasági és egyéb adatokat tesz elérhetővé, ezek használatával jelentősen lehetne pontosítani a program általi becslések elvégzését.

[58] pytest: helps you write better programs (2022) https://pytest.org/ (Utolsó letöltés: 2022.11.15.)
[59] Farkas Gábor - A TDD (Test Driven Development) világa, I. rész: Bevezetés (2015.05.14 https://ithub.hu/blog/post/A_TDD_vilaga_I_resz_Bevezetes (Utolsó letöltés: 2022.11.15.)

- Az adatok megjelenítése is jelentős mértékben átalakításra szorulna, lehetne komplex, akár több országot, vagy mutatót is egy-egy komolyabb, akár interaktív grafikonon is ábrázolni.
- Hasznos lenne a program kezelése szempontjából egy egyszerű grafikus kezelőfelületet alkalmazni, mely a parancssorban kevéssé jártas felhasználóknak is biztosítaná a program használatát.
- Helyi adatbázissal lehetne javítani az adatok feldolgozásának a sebességén, akár rendszeres adatszinkront végezve, ahogy például valamilyen ellenőrző megoldással az adatok hitelességét is nyomon tudnánk követni. (Itt nem a forrásadatok hitelességéről, hanem az internetes adatfolyamon keresztül letöltött adatok hitelességéről van szó.)
- Új típuselemzések esetén ezek hermeneutikai moduljának kialakítása

6 ÖSSZEFOGLALÁS

Hallgatótársaimmal együttműködve sikerült a felsőoktatás egyik legfontosabb értéke alapján csapatban elérnünk a kitűzött célunkat. A fentebb vázolt program ennek csak egy része volt, ami önmagában a többiek munkája mellett Horváth Géza, Pitlik László, Rikk János mentorálása nélkül nehezen valósulhatott volna meg ilyen szűkös időkeret alatt. Ez úton is köszönjük a támogatásukat!

Dolgozatommal igyekeztem rávilágítani, hogy kellően kis részekre felbontva egy komplex problémát, lépésről-lépésre haladva milyen mértékű előrehaladást lehet elérni a manuálisan végzett adatgyűjtés, adatszűrés, kiértékelés és megjelenítés helyett automatizáltan.

A Python nyelv adta rugalmasság, platformfüggetlenség révén a kódból akár binárist is tudnánk fordítani az elterjedt operációs rendszerekre, így akár előbb-utóbb egy kész, piacképes termék is megvalósulhatna a becslések kiértékelésére.

FELHASZNÁLT FORRÁSOK JEGYZÉKE

Christophe Benz - Introduction to DBnomics in Python (2018.10.26.) https://notes.quantecon.org/submission/5bd32515f966080015bafbcd (Utolsó letöltés: 2022.11.15.)

Julien Danjou (2018.12.) Serious Python - Black-Belt Advice on Deployment, Scalability, Testing, and More. San Francisco, No Starch Press. ISBN-13 9781593278786

Frank Hutter, Lars Kotthoff, Joaquin Vanschoren (2019) Automated Machine Learning Methods, Systems, Challenges. Springer Nature Switzerland AG. ISBN 978-3-030-05318-5 (eBook) https://link.springer.com/book/10.1007/978-3-030-05318-5 (Utolsó letöltés: 2022.11.15.)

Koki Noda – How to use OECD data in Python (2022.02.11.) https://medium.com/@koki_noda/how-to-use-oecd-data-in-python-69a0234c6b27 (Utolsó letöltés: 2022.11.15.)

Pitlik Marcell – Térfogati égés optikai vizsgálata és mesterséges intelligencia alapú elemzése, 7. MELLÉKLETEK, pp 49-55. (2020) http://www.combustioninstitute.hu/tdk_dij/2021/PitlikMarcell_TDK2020.pdf (Utolsó letöltés: 2022.11.15.)

Robot-polgár (e-gazság projekt 2022 vs 2010) (2022) https://miau.my-x.hu/miau/285/e_gazsag_2022/ (Utolsó letöltés: 2022.11.15.)

Arthur Turrell - Coding for Economists (2022) https://aeturrell.github.io/coding-for-economists/data-extraction.html (Utolsó letöltés: 2022.11.15.)

Választás 2010 Magyarország, e-gazság projekt (2010) https://miau.my-x.hu/myx-free/bevezetes.html (Utolsó letöltés: 2022.11.15.)

Választás 2022 Magyarország – avagy Robotpolgár 2022 (Election 2022 Hungary or Robot-Citizen 2022) (2022) https://miau.my-x.hu/miau/281/election_2022_hu.docx (Utolsó letöltés: 2022.11.15.)

MELLÉKLETEK

1. számú melléklet

A teljes forráskód

Mivel nem egyetlen, kisméretű fájlból áll a program forrása, így egy Zip fájlba csomagoltam be a teljes forrást (Python virtualenv előkészítéssel és dokumentációval együtt).

Ez alapján az olvasó a saját számítógépén is képes lehet futtatni a programot. Ehhez Python 3.10-et használtam, Windows alatt a WinPython[60], Linux és MacOS alatt az adott környezet szerinti Python 3.10+ telepítése után „virtualenv” [61] segítségével célszerű az aktiválást követően a függőségeket telepíteni (például „pip -r requirements.txt”), majd futtatható az „e_gazsag_lite.py” fájl.

Szakdolgozat-GazdMen-program-e_gazsag_lite-v1.0.zip

PDF-be átkonvertálás esetén elveszik a fájl, így a következő webes tárhelyen is megosztottam azt:

https://miau.my-x.hu/miau/291/e-gazsag_2022/Szakdolgozat-GazdMen-program-e_gazsag_lite-v1.0.zip

[60] WinPython (2022) https://winpython.github.io/ (Utolsó letöltés: 2022.11.15.)

[61] Python virtualenv (2022) https://realpython.com/python-virtual-environments-a-primer/ (Utolsó letöltés: 2022.11.15.)