TDK-dolgozat

Török Gergő

Exobolygók automatizált detektálása gépi tanulási módszerekkel

Török Gergő
III. évf. programtervező informatikus

Témavezető: Dr. Tóth László tudományos főmunkatárs

SZTE TTIK Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszék

Összefoglaló

A Naprendszerünkön kívüli élet felfedezése izgalmas kihívást jelent a kutatók számára. Ehhez elengethetetlen az exobolygók detektálása, melyek otthont adhatnak az élet számára, sőt akár magának az emberiségnek is a jövőben. Az asztrofizikusok űrteleszkópok segítségével vizsgálják a Tejútrendszer csillagait, melyek fényességváltozásából további égitesteket fedezhetnek fel. A mérések osztályozása (aszerint, hogy milyen jelenség következtében történt fényességcsökkenés), egy igen munkaerő-igényes feladat. Az asztrofizikában jelenleg hagyományos algoritmusalapú megközelítéseket használnak az adathalmazok feldolgozására, amelyeket utána egyesével vizsgálnak át a csillagászok. A gépi tanulás nagy mértékben felgyorsíthatja ezt a folyamatot. A dolgozatban először a Kepler űrtávcső méréseiből készült adathalmazt használom klasszikus osztályozók és egy neurális háló tanítására, majd kiértékelésére. A TESS (Transiting Exoplanet Survey Satellite) űrtávcső példátlan mennyiségű fotometriai megfigyelést készít. A dolgozatban mélytanulással osztályozom a TESS méréseiből készült adatokat a Liang Yu etal. által publikált módszer és neurális hálózat módosított változatával. A metaparaméterek finomhangolásának felgyorsításához egy szűrőeljárást készítettem az adathalmazhoz. Eredményeimet összevetem a szakterület korszerű kutatási eredményeivel, és röviden kifejtem azok hiányosságait. A modellem 97%-os accuracy-vel osztályozza a fénygörbéket, és 97,5%-os recall érték mellett 36,0%-os precision értékkel képes exobolygókat azonosítani.

Kulcsszavak: exobolygó, tranzit, fénygörbe, osztályozók, mélytanulás

A dolgozat során használt Colab füzeteim és modelljeim elérhetősége:

https://github.com/gergotorok02/Exobolygo_detektalas-1

Tartalomjegyzék

Bevezetés.....4
1. Távoli égitestek felfedezése.....4
2. Tranzit.....5
3. Kepler misszió.....6
A Kepler űrtávcső adatai.....7
1. A Caltech által készített adathalmaz.....7
2. Az adatok előkészítése a tanításhoz.....8
3. Modellek.....8
3.1. Logikai regressziós osztályozó.....8
3.2. K-legközelebbi szomszéd osztályozó.....9
3.3. Döntési fa osztályozó.....10
3.4. Véletlen erdők osztályozó.....11
3.5. Neurális hálózat.....11
4. Értékelés.....12
TESS.....13
1. A TESS űrszonda.....13
2. Fogalmak áttekintése.....13
3. TESS adatok.....15
4. Fénygörbék osztályozása.....17
5. Gépi tanulás.....19
6. Triage.....24
7. Vetting.....24
7.1. Célkitűzés.....24
7.2. Az adatok parszolása.....25
7.3. Kiegyensúlyozatlanság kezelése.....25
7.3.1. SMOTE.....25
7.3.2. Adataugmentáció.....26
7.3.3. Adatok szűrése a paraméterek és metaparaméterek hangolásához.....27
7.3.4. Kiegyensúlyozott batch generálás.....28
7.4. A neurális hálózat.....28
7.5. A modell tanítása.....30
7.6. A modell kiértékelése.....30
8. Kritikai észrevételek Yu, L. et al. (2019) eredményeivel kapcsolatban.....32
8.1. Adatszivárgás.....32
8.2. Robosztusság, keresztvalidáció.....33
9. TESS összegzés.....35
10. További irányok és kitekintés.....35
Irodalomjegyzék.....37
Appendix.....40
1. A dolgozathoz felhasznált mesterséges intelligencia alapú eszközök.....40
2. A dolgozat során használt Colab füzeteim és modelljeim.....40

Bevezetés

1. Távoli égitestek felfedezése

Percival Lowell marsi élet után kutatva 1902-ben felvetett egy elméletet, mely szerint létezik egy 9. bolygó a Naprendszerünkben. 1910-ben vásárolt egy “blink comparator” nevű szerkezetet a “Planet X” nevű programjához. Az eszköz lényege, hogy két képet tud nagyon gyorsan váltogatni, így könnyebbé téve a két kép közti különbségek, eltérések észrevételét. (https://lowell.edu/discover/history-of-pluto/)

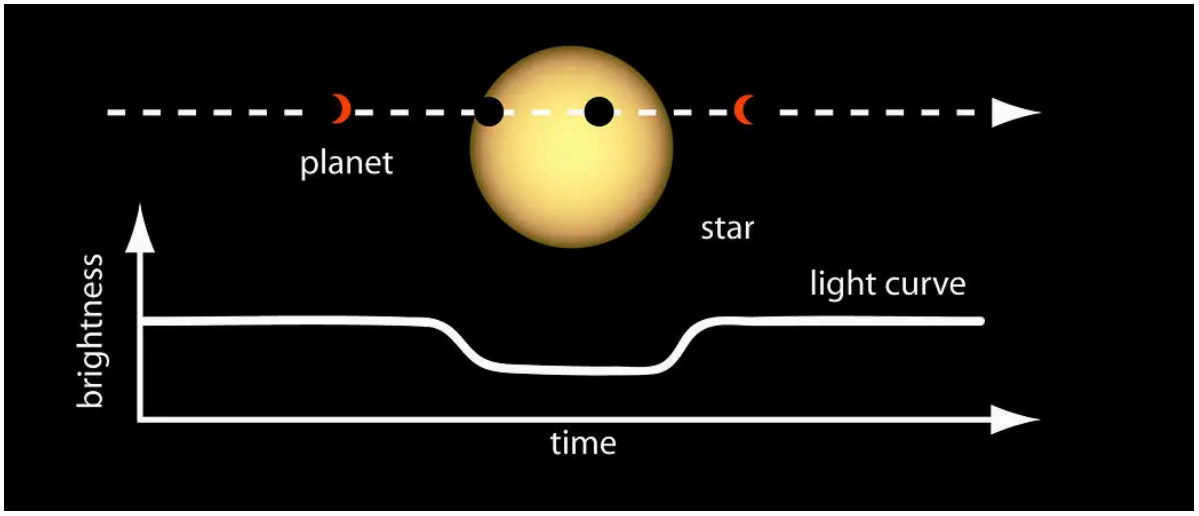
Az ókorban már felismerték, hogy bár az égbolt csillagai látszólag mozdulatlanok, néhányuk mégis rendszeresen vándorol az égen. Ezeket az égitesteket, amelyeket ma a Naprendszer bolygóiként ismerünk, a görögök "planétészeknek", azaz vándorlóknak nevezték el. Az arizonai Lowell Obszervatóriumban dolgozó Clyde Tombaugh csillagász módszere ehhez a megfigyelési technikához hasonlított: éjszakai felvételeket készített az égboltról különböző napokon ugyanarról a területről, és megkereste azokat a fényes pontokat, amelyek pozíciója megváltozott (kivéve azokat, amelyek már ismert bolygóinkhoz tartoznak), hogy így új mozgó objektumokat fedezzen fel a Naprendszerünkben.

1930. február 18-án, 19 évvel a blink comparator megvásárlása után, ezzel a technikával Clyde Tombaugh felfedezte a Pluto nevű kisbolygót.

Bár a fent említett megközelítés rendkívül hasznos a közeli objektumok vizsgálatára, más bolygórendszerek túlságosan távol vannak ahhoz, hogy az ottani égitesteket ilyen módon megfigyeljük. Amikor a saját Naprendszerünkben vizsgálódnak a csillagászok, akkor nem kizárólag a nagyságrendekkel rövidebb távolságok miatt tudnak blinkelésre alkalmas fényképeket készíteni. A Nap helyzetével ellentétes irányban vizsgálódnak, ezáltal egy sokkal homogénebb fényintenzitású szeletét látják az égboltnak. Ha egy távoli csillag bolygóit szeretnénk megvizsgálni, kénytelenek vagyunk “ránézni” az adott csillagra, melynek erős luminozitása ellehetetleníti a környezetében lévő, saját fénnyel nem rendelkező objektumok vizsgálatát.

2. Tranzit

Exobolygók felfedezésére a legelterjedtebb és egyben legsikeresebb módszer a tranzitok megfigyelése. A jelenleg ismert exobolygók túlnyomó többségét ezen elv szerint fedezték fel (https://science.nasa.gov/exoplanets/whats-a-transit/). Tranzitnak nevezzük azt a jelenséget, amikor a megfigyelő és a megfigyelt csillag között elhalad egy égitest. A Naprendszerünkön belül akkor lehet megfigyelni tranzitot, amikor a Merkúr vagy a Vénusz elhalad a Föld és a Nap között. A tranzitok azért segítenek a távoli bolygók felfedezésében, mert azok áthaladásukkor csökkentik a csillaguk megfigyelhető fényességét. Az ilyen jelenségeket fénygörbékkel ábrázolják, ahol az adott csillag fényességének értéke van feltüntetve az idő függvényében (1. ábra). Egy tranzit esetében a fénygörbe csökkenést fog mutatni. Ilyen események során a megfigyelt bolygó egyéb tulajdonságait is megismerhetjük, a tranzitok periodikus bekövetkezése alapján kiszámíthatjuk a bolygó keringési idejét, a csillag fényességének mért csökkenéséből pedig a bolygó méretét. Ezen felül például a fény szóródásából információkat nyerhetünk a bolygó légkörének összetételéről, azonban a dolgozatomban erre nem térek ki.



1. ábra

A NASA Ames által készített ábra egy bolygótranzitról. Az ábra a tranzit során bekövetkező fényességcsökkenést szemlélteti egy fénygörbe segítségével.
forrás: https://www.nasa.gov/image-article/light-curve-of-planet-transiting-star/

3. Kepler misszió

2009. március 6-án a NASA a Discovery-program keretében felbocsátotta a Kepler űrtávcsövet azzal a céllal, hogy exobolygókat fedezzenek fel a Tejútrendszerben. Az eszköz küldetése során kifejezetten olyan bolygókat keresett, amelyek az adott csillag lakható övezetében vannak, a víz folyékony halmazállapotban megtalálható rajtuk, és méretük a Föld méretének felétől a kétszereséig terjed. A három és fél évig tartó misszió során körülbelül százezer "main sequence" csillagot figyelt meg (olyan csillagok melyek fúziójuk során hidrogénből héliumot hoznak létre, tömegük a Nap tömegének tizedétől kétszázszorosáig terjednek) (https://science.nasa.gov/mission/kepler/).

2018. október 30-án, kilenc évvel a küldetés kezdete után, a NASA bejelentette, hogy a Kepler űrtávcső üzemanyaga elfogyott, így küldetése véget ért. Az űrtávcső hagyatéka 2600 bolygó felfedezése a Naprendszerünkön kívül, közülük sok kedvező lehet az élet számára.

A Kepler űrtávcső adatai

1. A Caltech által készített adathalmaz

A California Institute of Technology egyetem készített egy adathalmazt a Kepler űrtávcső méréseiből, és bárki számára hozzáférhetővé tették a NASA exobolygó archívumában. A dolgozatban először megvizsgálom az adatokat, majd gépi tanulási módszerekkel predikálok rajtuk. Ahogy azt nem sokára látni fogjuk, ez az adathalmaz rengeteg előfeldolgozáson esett át, így egyszerű osztályozók is szép eredményeket tudnak rajta elérni. Emiatt itt még nem fogunk mélyen belemenni az adatokba, csupán betekintést nyerünk abba, hogy gépi tanulással képesek lehetünk osztályozni a méréseket aszerint, hogy valóban exobolygót találtak-e. Ennek a fejezetnek a lényege csupán az egyszerű osztályozók áttekintése, a komolyabb exobolygó detektálás a további fejezetekben kerül elő.

Az interneten fellelhető számos szép eredmény, amelyeket ezen adathalmaz felhasználásával értek el. A fejezet másik fő aspektusa, hogy megmutassuk, hogy ezek az eredmények nem teljes mértékben relevánsak. Az adathalmaz olyan gondosan szelektált és előkészített, hogy primitív technikákkal is magas metrikákat lehet elérni. Ez pont az ellentettje a nyers, új adatokon történő exobolygó detektálásnak.

Az adathalmaz 82 oszlopot tartalmaz, paraméterei például a “KepID”, amely azonosítja az adott mérést, “KOIName”, amelyet azon mérések kapnak, amelyeken feltehetőleg valamilyen áthaladás történt. Az utólag, csillagászok által is exobolygónak tekintett objektumok pedig nevet is kapnak, ezeket a “KeplerName” oszlopban találhatjuk. Természetesen azon mérések esetében, amelyeknél nem találtak exobolygót, NaN érték szerepel. A “DispositionUsingKeplerData” oszlopban feltüntették, hogy mely mérések lehetnek exobolygók (ezek “CANDIDATE” címkével jelöltek), illetve téves mérések (“FALSE POSITIVE”). Az “ExoplanetArchiveDisposition” oszlop pedig ehhez nagyon hasonló, viszont itt azok az objektumok, amelyek korábban “CANDIDATE” címkével szerepeltek, de azóta felvettek a hivatalos exobolygók listájára (megtalálhatóak a NASA archívumában) “CONFIRMED” címkét kaptak. Ezek lesznek az osztálycímkéink. A tanulást és a kiértékelést két részre osztom. Az egyik esetben az előbbi oszlop adatait használom, vagyis a “CANDIDATE” és “FALSE POSITIVE” osztályokba próbálom sorolni a méréseket. A másik esetben pedig a “CONFIRMED” és “FALSE POSITIVE” címkéket használom. Előkerülnek olyan tulajdonságok is, mint az “OrbitalPeriod”, “TransitEpoch”,

“TransitDuration”, “TransitDepth” stb. amelyek a dolgozat későbbi részében rendkívül fontosak lesznek, és TESS adatok előfeldolgozásánál mélyebben tárgyalom majd ezeket.

2. Az adatok előkészítése a tanításhoz

Egyes oszlopok nem szolgálnak tanulásra használható adatokkal. Például a korábban említett “KeplerName” vagy a “KOIName” csupán egy emberek által adott elnevezése egy mérésnek. Az ilyen oszlopokat eltávolítom az adathalmazból. Ezután eltávolítom azokat a sorokat, amelyek NaN értékeket tartalmaznak. Ezek után 5426 sor és 57 oszlop marad. Az osztályok eloszlása mindkét esetben kellően kiegyensúlyozott, így ezzel nem foglalkozom. Ez is egy olyan tulajdonság, amely általánosságban nem jellemző exobolygók detektálásakor, és nagyobb hangsúlyt fog kapni a TESS adathalmaznál. A korábban említett osztálycímkéket enkódolom, és tanító, illetve teszthalmazokra bontom az adathalmazt.

3. Modellek

Klasszikus osztályozó modelleket fogok használni a predikálásokhoz, illetve végül egy egyszerűbb neurális hálót is kipróbálok.

3.1. Logikai regressziós osztályozó

Először logikai regressziós modellel kísérletezem. A logikai regresszió alkalmas lehet abban az esetben, amikor két osztálycímkénk van és az adatok folytonosak. Ez a két feltétel jelen esetben fennáll. A modell egy “S görbét” próbál illeszteni, amivel szeparálja a két osztályt (Ruczinski et al., 2003). A görbe képlete:

$$P(Y=1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n)}}$$

ahol P(Y=1) a valószínűség, hogy a célváltozó (Y) értéke 1, azaz a vizsgált esemény bekövetkezik, *e* az Euler-szám. β_0 ,β_1 ,...,β_n a regressziós együtthatók, amelyeket az

adatokon való tanítás során határoznak meg. Ezek az együtthatók határozzák meg, hogy az egyes bemeneti változók (X_1, X_2, ..., X_n) milyen mértékben befolyásolják a célváltozó log-odds-át. X_1, X_2,..., X_n a bemeneti változók (vagy jellemzők), amelyek alapján meg szeretnénk becsülni a célváltozó értékét. A gyakorlatban a "scikit-learn" nevű rendkívül népszerű Python modult fogom használni valamennyi modellnél (Kramer, O. et al., 2016). A csatolt kódban látható, hogy csak 78% accuracy értéket ért el a modell, mely talán szembemegy az eddig említett "egyszerű modellekkel is magas értékeket érhetünk el" elvvel. Azonban a logikai regresszió egy rettenetesen primitív modell, és alkalmas arra, hogy szemléltessem, az adathalmaz nem nevetségesen egyszerű, egy ilyen megközelítés még nem tud kellően hatékony lenni. A következő modellem kis mértékben javítani fog az eddigi eredményen, de a valódi változást majd a harmadik modell fog eredményezni.

3.2. K-legközelebbi szomszéd osztályozó

A "K-legközelebbi szomszéd", egy széles körben ismert egyszerű osztályozó. Az algoritmus működése a következő: meghatároz egy K számot, amely a legközelebbi szomszédok számát jelöli (Peterson, L. E. et al., 2009). Amikor egy új mintapontot kell osztályozni, megkeresi a K legközelebbi szomszédjait a tanítási adathalmazban, figyelembe véve az összes pont közötti távolságot. Ezt követően osztályozásra kerül sor a K legközelebbi szomszédok között előforduló leggyakoribb címke alapján. Az eljárás három fő paramétere a szomszédok száma, a távolság számításához használt metrika és a súlyok. A paraméterek beállításához úgynevezett "grid search" eljárást használok, vagyis mind a három paraméterhez felsorolok több értéket, és a modell kiértékeli az adatokat minden lehetséges kombinációra. A szomszédok számát 3, 5, 7, 10 és 12 értékekre tesztelem, távolsági metrikáknak euklideszi-, Manhattan-, Csebisev- és Minkowszki-távolságot állítok be, súlyoknak pedig "uniform" és "distance" súlyokat. ("unform" esetében minden szomszéd egyenlő súllyal járul hozzá a az eredmény kiszámításához, "distance" pedig a korábban említett távolságokkal arányosan.)

A csatolt Colab füzetben látható, hogy a kiértékelés után a modell 82,5%-os accuracy értéket ért el, a paraméterek további finomhangolásával valószínűleg lehetne javítani néhány százalékot, azonban érdemes kipróbálni más modelleket, amelyekkel magasabb értékek érhetők el.

3.3. Döntési fa osztályozó

A Döntési fa osztályozó egy jóval erősebb eszköz lehet ennél a feladatnál. A módszer lényege, hogy egy döntési fa modellt épít fel, amely a jellemzők értékei alapján hoz döntéseket, vagy jósol címkéket az adatpontokra. A döntési fa alapvetően egy bináris fa, ahol minden belső csomópont egy jellemző tesztjét reprezentálja (pl. egy jellemző értéke nagyobb-e, mint egy küszöbérték), minden él a teszt egy lehetséges eredményét reprezentálja, és minden levél csomópont egy osztálycímkét, vagy egy regressziós értéket tartalmaz. A fa építésének lépései általában a következők: Kiválasztjuk a legjobb jellemzőt, ami alapján elválasztjuk az adathalmazt. Ez általában Gini-index vagy információnyereség alapján történik. Az információnyereség a szülőhalmaz entrópiájának és a jellemző alapján létrehozott részhalmazok súlyozott entrópiájának különbsége. Az entrópia egy statisztikai mérték, amely az osztályok közötti tisztaságot vagy rendezetlenséget méri.

$$IG(D,a) = H(D) - \sum_{v \in \text{Értékek}(a)} \frac{|D_v|}{|D|} H(D_v)$$

ahol *IG(D,a)* az információnyereség a *D* adathalmazra és az '*a*' jellemzőre. *H(D)* az entrópia a *D* teljes adathalmazban. *Értékek(a)* az '*a*' jellemző lehetséges értékeinek halmaza. $|D_v|$ a *v* értékkel rendelkező elemek száma az '*a*' jellemzőben. $|D|$ az összes elem száma a *D* adathalmazban. $H(D_v)$ az entrópia azon elemekre, amelyek az '*a*' jellemzőben a *v* értéket veszik fel. A Gini-index pedig egy mérték, amely azt számolja, hogy véletlenszerűen kiválasztott elem milyen valószínűséggel lenne rosszul címkézve, ha véletlenszerűen választanánk egy címkét a részhalmazból.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

ahol *Gini(D)* a Gini-index a *D* adathalmazra, *m* az osztályok száma és p_i az i-edik osztály relatív gyakorisága a *D* adathalmazban. Az algoritmus második lépése, hogy létrehozunk egy döntési pontot ebben a jellemzőtérben a kiválasztott küszöbérték alapján. A harmadik pedig, hogy rekurzívan ismételjük ezt a folyamatot minden részhalmazra, amíg minden adatpontot jól nem tudunk osztályozni, vagy el nem érjük az előre meghatározott mélységi

limitet. A csatolt Colab füzetben látható, hogy a kiértékelés után a modell 97,3% accuracy értéket ért el hasonlóan magas recall, F1-score és precision mellett.

3.4. Véletlen erdők osztályozó

Az utolsó egyszerűbb osztályozó, amelyet kipróbáltam, a Véletlen Erdők. Ez az osztályozó az “ensamble learning” kategóriába tartozik, amelynek alapötlete, hogy több kisebb modellt kombinálva hozunk létre egy erősebb predikációs modellt. Az algoritmus menete: Az eredeti adathalmazból többszörösen véletlen mintákat veszünk, amelyeket a különböző döntési fák tanításához használunk fel. Egy ilyen mintavételt “bootstrap”-nek nevezünk. Minden bootstrap mintára egy döntési fát építünk. A fa építése során minden döntési pontnál csak a jellemzők egy véletlenszerűen kiválasztott altartományát vesszük figyelembe. Ez növeli a fák közötti diverzitást és csökkenti a túlilleszkedés kockázatát. Amikor a modellnek egy új mintát kell osztályoznia, minden fa az adathalmazban ad egy "szavazatot". Osztályozási feladatok esetében a leggyakoribb szavazatot kapott osztály lesz a végeredmény. A K-legközelebbi szomszéd mintájára itt is gridsearch-öt alkalmazunk. A csatolt Colab füzetben látható, hogy a kiértékelés után a modell 98,47%-os accuracy értéket ért el.

3.5. Neurális hálózat

Végül kipróbáltam egy egyszerűbb neurális hálót. Egy 128 neuron szélességű bemeneti réteget (relu aktivációval, és 0.5-ös dropouttal) és egy 64 neuron szélességű rejtett réteget (szintén relu aktivációval és 0.5-ös dropouttal) használtam. A kimeneti réteg csupán egyetlen neuront tartalmaz sigmoid aktivációval. Adam optimalizálót használtam, veszteségfüggvénynek pedig bináris kereszt-entrópiát. A tanítóhalmaz 10%-át validációs halmaznak leválasztottam, a modellt 10 epochig tanítottam 32-es batch mérettel. A modell 98,43%-os accuracy-t ért el.

4. Értékelés

Korábban említettem, hogy két feladatot definiáltam, azonban a két feladat során elért eredmények lényegében megegyezőek voltak, így az egyes pontokban nem említettem ezeket külön-külön. A csatolt Colab füzetben látható, hogy egy Véletlen Erdők osztályozóval és egy abszolút nem optimalizált neurális hálóval is 98%-nál magasabb accuracy-t lehetett elérni (hasonlóan magas precision, recall, F1-score metrikák mellett). Emiatt ezt az adathalmazt nem tekinthetjük túl nagy kihívásnak. Az adatok minimális előfeldolgozása elég radikálisan történt részemről, ez az adathalmaz épp ilyen apró kísérletekre alkalmas. A következőkben áttérek a TESS adatok feldolgozásához, és az azon való predikciókhoz, ami a dolgozat fő fókusza.

TESS

1. A TESS űrszonda

2018. április 18-án a NASA elindította a TESS űrszondát (2. ábra), melynek célja exobolygók felfedezése. Eredeti küldetésén túlmenően egyéb fénycsökkenéssel járó jelenségek következtében aszterodiákat, pulzáló csillagokat és távoli galaxisok szupernóváit is vizsgálja. A küldetés különlegessége, hogy a mai napig zajlik, és folyamatosan friss adatokat küld az űrszonda, amelyeket a NASA feltölt az internetre. Emiatt valóban eddig ismeretlen exobolygókat is detektálhatunk, ha sikerül egy megfelelő modellt készítenünk (https://science.nasa.gov/mission/tess/).

2. ábra

A TESS űrszondáról készült 3D render.

2. Fogalmak áttekintése

fénygörbe: egy csillagászati objektum fényességének időbeli változását mutatja meg grafikonon. Ezen görbe segítségével a csillagászok és kutatók elemezhetik, hogyan változik egy csillag, vagy más égitest fényereje az idő folyamán. A fénygörbék különösen fontos szerepet játszanak a változócsillagok vizsgálatában és az exobolygók keresésében (https://imagine.gsfc.nasa.gov/features/yba/M31_velocity/lightcurve/lightcurve_more.html).

epoch: egy adott referencia-időpontot jelent az adatokban. Csillagászati kontextusban az epoch általában azt a pillanatot jelöli, amikor először megfigyeltek egy jelenséget, vagy amikor az adatgyűjtési sorozat kezdődött. Például ha egy exobolygó tranzitjait figyeljük meg, az epoch az első tranzit időpontja (Soop 1994).

periódus: az az időtartam, ami alatt egy ismétlődő esemény (pl. egy exobolygó tranzitja) teljes ciklust fut be. Más szóval, az az idő, amely után az adott jelenség pontosan ugyanabban az állapotban ismétlődik (Montenbruck, O. et al., 2000). A TESS adatokban a periódus kritikus fontosságú, mert segít megjósolni, hogy mikor várható a következő tranzit, amely kulcsfontosságú az exobolygók megerősítéséhez.

hajtogatott fénygörbe: olyan technika, amelyet az ismétlődő asztronómiai jelenségek, mint például az exobolygó tranzitok megfigyelésére használnak. Ennek lényege, hogy az összes elérhető tranzitadatot egyetlen perióduson belülre "hajtogatják", így a tranzitokból származó információk egymásra rakódnak. Ezáltal jobban láthatóvá válnak a kisebb jelenségek is, amelyek egyetlen tranzit során talán nem lennének észlelhetők. A fénygörbe hajtogatása során a csillag fényességének méréseit úgy rendezik át, hogy a periódus minden egyes pontján az összes tranzit megfigyeléseit összeadják vagy átlagolják (https://imagine.gsfc.nasa.gov/science/toolbox/timing2.html).

fedési kettőscsillagok: olyan kettőscsillag-rendszerek, ahol a két csillag a pályájukon mozogva egymás előtt halad el, így időszakosan fedik egymást a Földről nézve. Ezek a csillagok egy olyan síkban mozognak, amely körülbelül egybeesik a megfigyelő látóirányával, így a két csillag periodikusan áthalad egymás előtt, ami fényességváltozást okoz (https://heasarc.gsfc.nasa.gov/docs/RXTE_Live/eclipse.html).

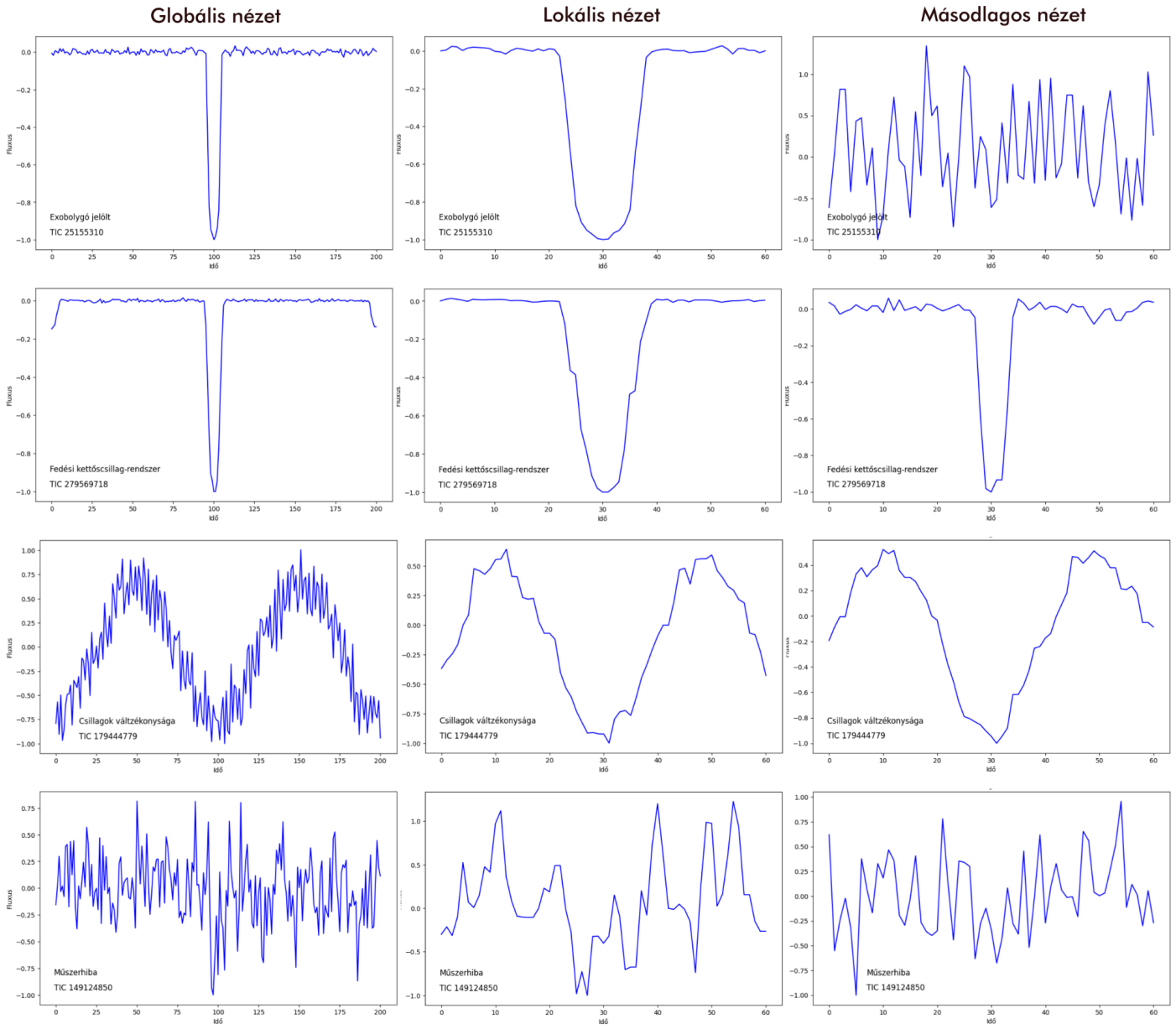
A globális, lokális és másodlagos nézetek fogalma (global, local, secondary view) nem teljesen egyértelmű. Klasszikus értelemben a globális nézet a teljes mérés időtartamát magába foglalja, általában több tranzitot is tartalmaz. Azonban a TESS adatoknál a rövid megfigyelési idők miatt ezt a nézetet nem szokták használni exobolygók keresése során. A dolgozatban Liang Yu et al. 2019 elnevezéseihez igazodom, ahol a globális nézet a teljes mérés hajtogatott fénygörbéje (valójában ez lenne a lokális nézet), a lokális nézet pedig ennek egy “felnagyított” verziója, ahol rövidebb időintervallumot választunk, és ezáltal elnyújtjuk vízszintesen a globális nézetünket. Így egy közelebbi képet kaphatunk a tranzitnak vélt eseményről, és finomabb jellemzőket azonosíthatunk. A másodlagos nézet azt az időintervallumot öleli fel, amikor az exobolygónak a “túloldalon” kellene lennie, vagyis a periódusa felénél jár. Ennek vizsgálata különösen hasznos lehet a fedési kettős jelenségek, azaz a kettőscsillag-rendszerek felismerésére. Egy

kettőscsillag-rendszernél olykor a másodlagos nézet esetében is tranzitot figyelhetünk meg, hiszen ilyenkor a második csillag halad el az első előtt. A fedési kettőscsillagok globális és lokális nézete rendkívül hasonló az exobolygókéhoz, így megnehezítik az osztályozást. A másodlagos nézet leginkább ezek kiszűrésére szolgál. A dolgozat további részében ezt a három fogalmat a fent említett értelemben fogom használni.

3. TESS adatok

A TESS adatok lekéréséhez a “lightkurve” Python modul egy kézenfekvő eszköz, amellyel könnyen hozzáférhetünk a NASA fénygörbe adataihoz mind a Kepler, mind a TESS űrteleszkópok esetében (Cardoso et al., 2018). A nyers TESS adatokat először egy pipeline feldolgozza és azután kerülnek föl a rendszerbe. Minden mérésnek van egy TIC ID száma (TESS Input Catalog), amellyel az egyes méréseket tudjuk azonosítani. A “search_lightcurve” függvénnyel egy adott TIC ID-hez tartozó adatokat tudjuk lekérni. Ezen felül további argumentumai is vannak, szűrhetünk például egy szektorra, ami az égbolt egy vizsgált területére, illetve egy “author”-ra, ami az adott fénygörbét feldolgozó pipleline-ra utal. Ilyen pipeline például a SPOC (Science Processing Operation Center) a NASA Ames Kutatóközpont, a QLP (Quick Look Pipeline) ami a Massachusetts Institute of Technology egyetem, és a MAST (Mikulski Archive for Space Telescopes) a Mikulski Archívum adatfeldolgozó pipeline-jai. Az így megszerzett adatot a “download()”, vagy “download_all()” (ha több adat is elérhető) függvénnyel tudjuk letölteni. A kapott adatsorozatunk egy fénygörbe. Ha kirajzolunk egy ilyen fénygörbét, akkor láthatjuk, hogy miként változik a csillag felől érkező fényerősség (ez alapján tudunk majd tranzitokat keresni). Ha ismerjük a vizsgált objektum periódusát és a mérés epoch értékét, akkor elkészíthetjük a fénygörbe globális nézetét a “fold()” függvény segítségével. Egy ilyen letöltött fénygörbe rengeteg pontot tartalmaz, így nagy mennyiségű fénygörbe feldolgozásánál gyorsan kezelhetetlen méretű adatunk lesz. Ennek elkerülése érdekében a “bin()” függvénnyel megadhatjuk, hogy hány ponttal szeretnénk reprezentálni az adott fénygörbét. A metódus az egymást követő adatpontokat időintervallumokba sorolja. Minden egyes időintervallum az adatpontok átlagát vagy mediánját (a megadott módszer alapján) képviseli, amelyek az adott időintervallumra esnek. A lokális nézethez ismernünk kell a tranzit időtartamát, és annak csökkentésével “ránagyíthatunk” a fénygörbe középső részére. Ehhez a “truncate()” függvényt használom, amely a globális nézet középpontjától balra és

jobbra egyenlő időtartamú szakaszt vág ki. A másodlagos nézet esetében az epoch értékhez hozzáadjuk a periódus felét, így megkapjuk a másodlagos nézet vélt középpontját.



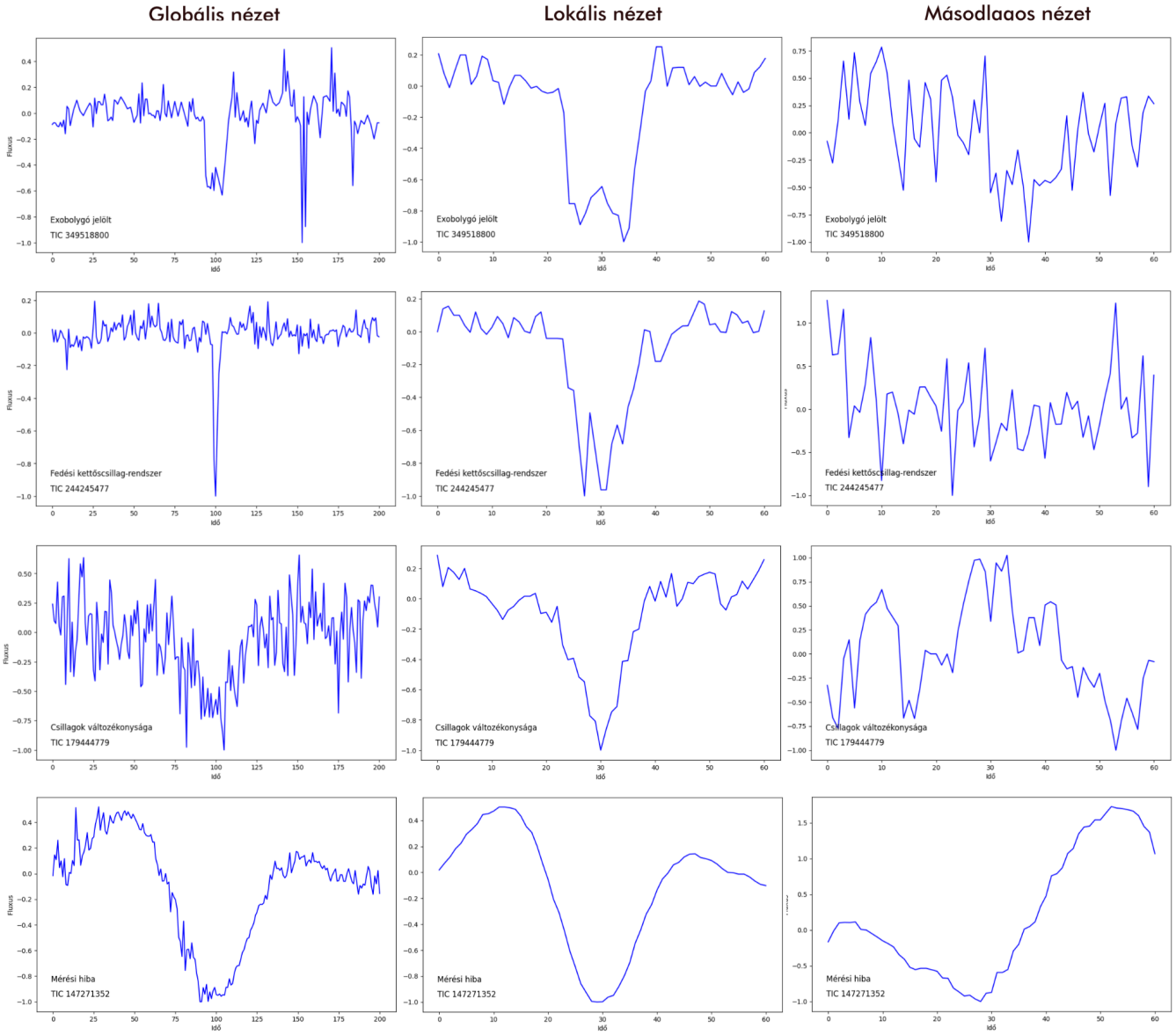
3. ábra. *Pár példa a fénygörbékből. Az első sorban egy exobolygó jelölt, a másodikban egy fedési kettőscsillag-rendszer, a harmadikban egy csillag változékonysága a negyedikben pedig műszerhiba fénygörbéi találhatóak.*

4. Fénygörbék osztályozása

A fénygörbék osztályozása elsőre egyszerűnek tűnhet. Csupán a globális nézetből megállapíthatjuk, hogy periodikus tranzit történik-e. Ezzel megkapunk minden tranzit eseményt, ahol vagy egy exobolygó haladt el, vagy egy kettőscsillag-rendszer egyik csillaga. Ezután a másodlagos nézet alapján szétválasztjuk ezt a két osztályt aszerint, hogy ott történik-e tranzit, ha igen akkor kettőscsillag-rendszert ha nem akkor pedig exobolygót találtunk. A probléma azt sejteti, hogy egyszerű matematikai eszközökkel, például görbe illesztéssel megoldható. Konstruálhatunk egy tipikus exobolygó globális, lokális és másodlagos nézetet, majd az ezektől vett távolságok alapján osztályozhatjuk a méréseket. Sajnos a valóságban ez ennél bonyolultabb. Először is nem csak tranzit válthat ki ilyen fénycsökkenést, hanem például napfoltok is előidézhetik, amelyek ráadásul rendkívül gyakoriak lehetnek, és a megfigyelés rövid ideje alatt periodikusnak is tűnhetnek. Szintén fényesség változással járnak a napkitörések és a pulzáló csillagok. Felléphet műszerhiba is, amely szintén előidézhet tüskeszerű csökkenéseket a fénygörbén. Ráadásul ezek olykor egyszerre is fellépnek, például egy exobolygó tranzit során történő napfolt kialakulása vagy megszűnése, kozmikus sugárzás, a Földről visszaverődő fény és még számos olyan tényező van, amely nem csak téves méréseket okoz, hanem a valós exobolygó tranzitok fénygörbéjét is torzítja. Olykor a pályahajlásszög, az excentricitás és eltérő csillagméretek miatt nem, vagy nagyon minimális tranzitszerű fénycsökkenés lép fel a fedési kettőscsillag-rendszerek másodlagos nézetében, így azok kiszűrése sem triviális. A TESS adatok ismertetésekor bemutatott fénygörbék válogatottan szép, elkülönülő példák voltak, azonban az adatok nagy részére ez sajnos nem igaz. A 4. ábra néhány tipikus példát mutat be.

Látható, hogy ezekben az esetekben már jóval nehezebb eldönteni, hogy melyik fénygörbe melyik kategóriába tartozik. Az első sorban egy exobolygó jelölt látható, azonban a globális nézetében feltűnően szerepel egy másik kiugró fényességcsökkenés is, ráadásul drasztikusabb is a tranzit által okozottnál. A másodlagos nézetében pedig pont középen történik fényességcsökkenés, azonban ilyenkor a bolygó a csillag ‘túloldalán” halad el, tehát nem kellene ilyen jelenséget tapasztalnunk. A második sor egy fedési kettőscsillag-rendszert reprezentál. A globális és lokális nézetek szépen elkülönülő tranzitot mutatnak, így az exbobolygó és a fedési kettőscsillag-rendszer lehetősége is fennáll. Ennek a sornak a másodlagos nézetében nem tapasztalunk tranzitot, így exobolygónak feltételezhetnénk. Az utolsó két sorban egy csillag változás és egy mérési hiba található, azonban olyan

karakterisztikus tulajdonságokat mutatnak, amelyek a tranzitokra jellemzők. Ráadásul a másodlagos nézetben nem tapasztalható középen kiugró fényességcsökkenés, így az exobolygó irányába húzhatja a predikciót.



4. ábra

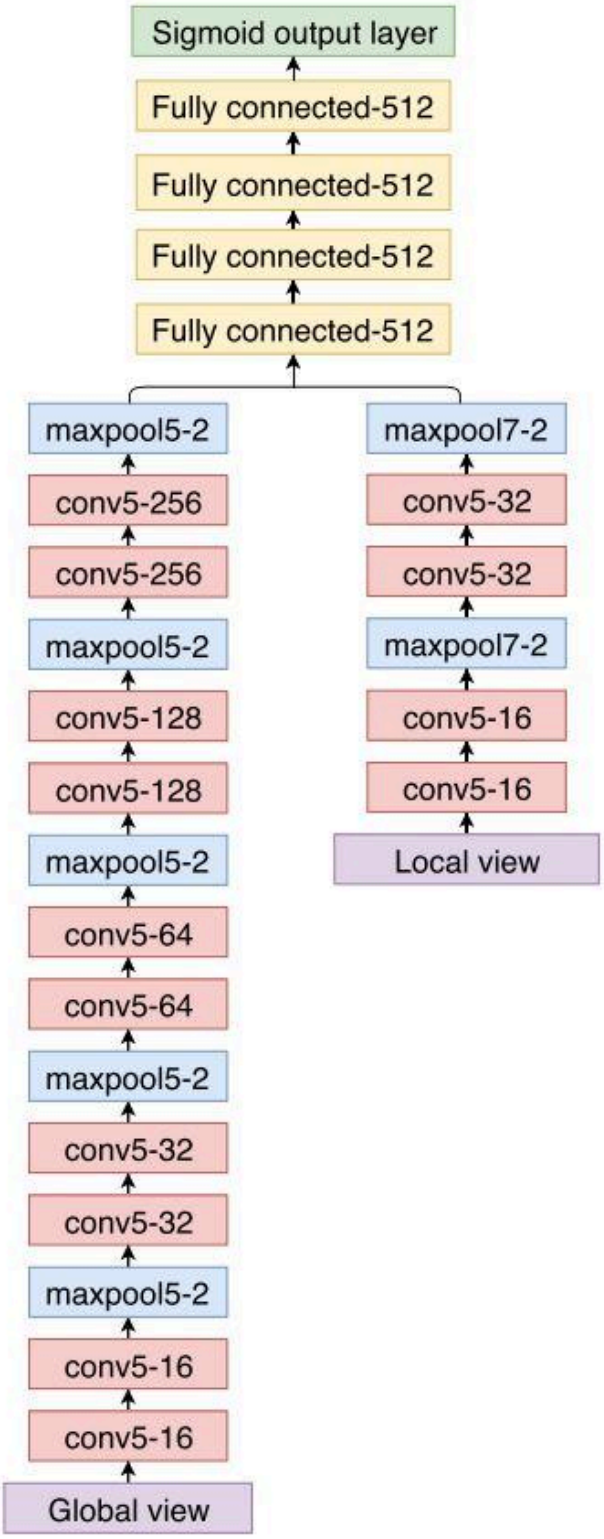
Néhány példa a fénygörbékből. Az első sorban egy exobolygó jelölt, a másodikban egy fedési kettőscsillag-rendszer, a harmadikban egy csillag változékonysága a negyedikben pedig mérési hiba fénygörbéi találhatóak.

Saját szerkesztésű ábra.

5. Gépi tanulás

Az exobolygók gépi tanulással történő keresésére irányuló kutatások kezdetben a fénygörbék klasszifikációjára egyszerű osztályozókat alkalmaztak. A komolyabb korai eredmények közé tartoznak például a Robovetter (Coughlin et al., 2016b), vagy az Autovetter (McCauliff et al., 2015) projektek, amelyek fa alapú modellt használtak a Kepler űrtávcső adatain. A későbbi kutatások egyre inkább áttértek a neurális hálózatok és a mélytanulás alkalmazására. Az egyik legkiemelkedőbb eredmény Shallue & Vanderburg (2018) kutatása volt, amely során megalkották az egyik legsikeresebb neurális hálót a Kepler adatokon történő exobolygó detektáláshoz. A modellt a mai napig sem tudták jelentős mértékben felülmúlni. Shallue és Vanderburg 2001 pontos globális nézetet és 201 pontos lokális nézetet használt bemenetként (Shallue, C. J., & Vanderburg, A. 2018). Az 5. ábrán látható módon a két fénygörbe külön konvolúciós szálon érkezik, majd végül több teljesen összekötött neuronréteggel kapcsolódnak össze. A konvolúciós neurális hálózatok képesek saját maguk kinyerni a szükséges jellemzőket a tanulási adatokból, az alacsony szintű jellemzőktől (mint például élek és szögek) a magas szintű jellemzőkig (mint például objektumrészek), ellentétben az egyszerűbb modellekkel, amelyek általában előre definiált jellemzőkön alapulnak. Továbbá képesek megtartani az input adatok térbeli strukturális információit, ami kritikus lehet olyan feladatoknál, amelyeknél az objektumok helyzete és elrendezése kulcsfontosságú, mint például képfelismerésnél vagy jelen esetben fényintenzitás időbeli változásánál.

Az “Identifying Exoplanets with Deep Learning. III. Automated Triage and Vetting of TESS Candidates” (Yu, L. et al., 2019) című publikáció az egyik legnagyobb előrelépés volt a TESS adatokon történő osztályozás témakörében. A kutatásban olyan rangos intézmények vettek részt, mint például az MIT (Massachusetts Institute of Technology), a The University of Texas at Austin, a Harvard asztrofizikai központja, és a NASA. A dolgozat ezen fejezetének további részében az ő általuk közzétett adatokat fogom én is használni, hogy az eredményeim összevethetőek legyenek az ő eredményeikkel, illetve más ezen adathalmazon elért eredményekkel. Az adatok a https://github.com/yuliang419/AstroNet-Triage és https://github.com/yuliang419/AstroNet-Vetting oldalakon érhetőek el.



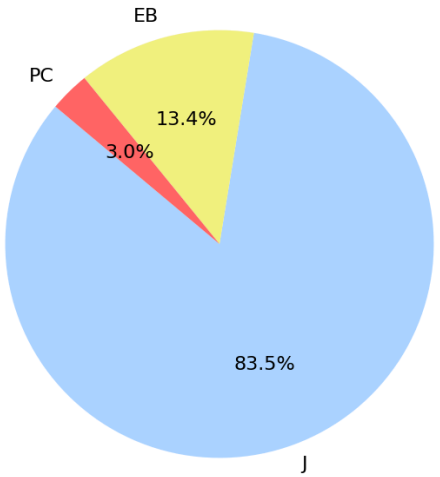
5. ábra.

Shallue és Vanderburg modellje.

Forrás: Yu, L. et al., (2019), p. 5.

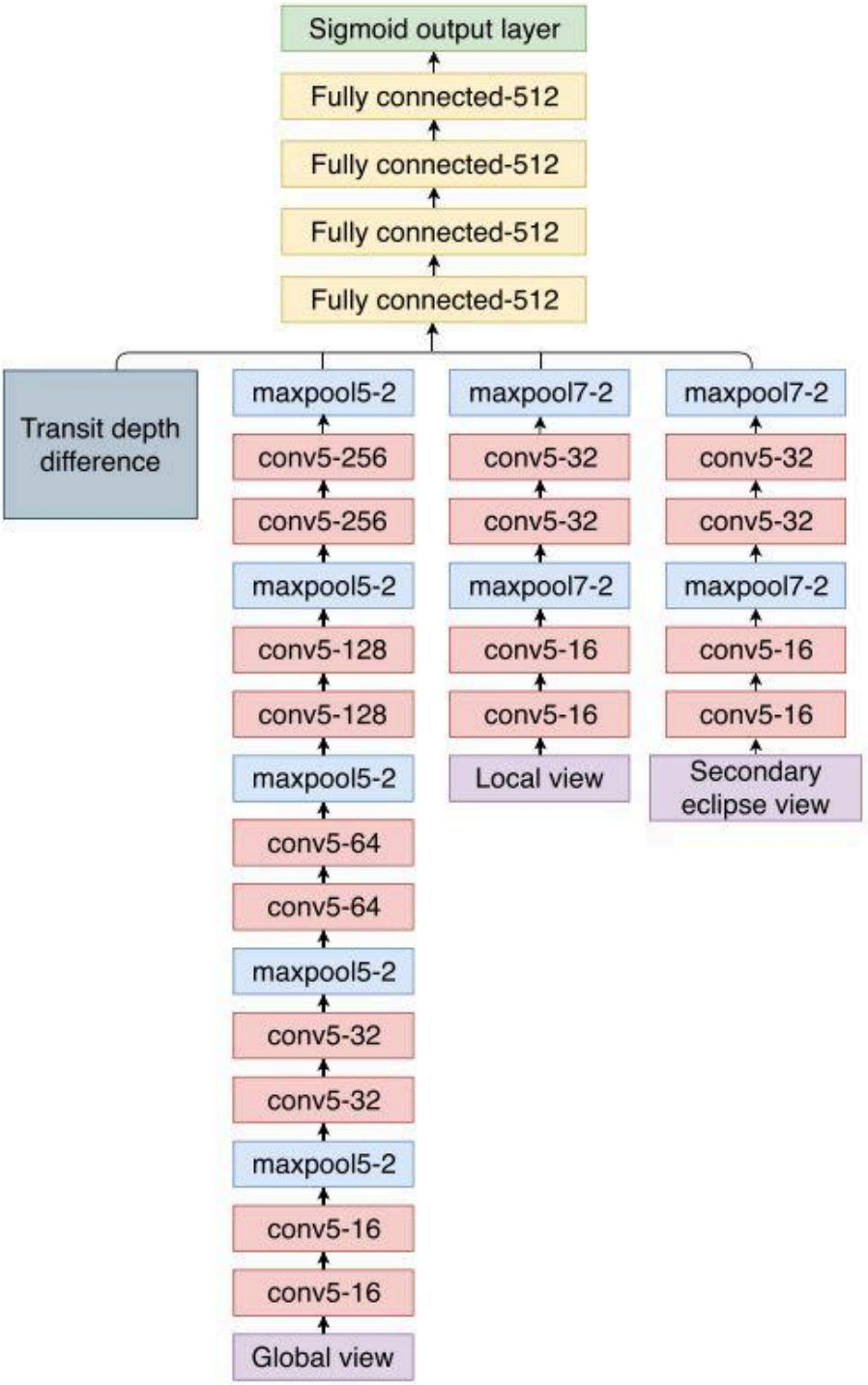
Az “Exoplanet detection using machine learning” (Malik, A. et al., 2022) című publikációban előkerült az az elv, mely szerint nem kellene teljesen elvetni a klasszikus osztályozókat, hiszen esetleg jobb teljesítményt nyújthatnak a mélytanulási megközelítéseknél. Abhishek M. és munkatársai TSFRESH nevű jellemzőkinyerő könyvtár segítségével 790 jellemzőt gyűjtöttek a fénygörbékről. Az eredményeiket összehasonlítom az általam elért eredményekkel a 7. pont után.

A Liang Yu és munkatársai által közzétett 16734 sort tartalmazó tces.csv fájl (Threshold Crossing Events, olyan események, amelyeknél fénycsökkenés történt) tartalmazza az első öt szektor méréseinek legfontosabb adatait: a TIC ID-t, az epoch értéket, a periódust, a tranzitnak vélt esemény időtartamát és egyéb adatokat, például, hogy melyik szektorhoz tartozik az adott objektum, melyik kamerával készült a felvétel stb. Az adathalmaz különlegessége, hogy egy szakértő egyesével minden sort felcímkézett aszerint, hogy exobolygó tranzit (PC, planet candidate), fedési kettőscsillag-rendszer tranzit (EB, eclipsing binary), csillagoban fellépő jelenségek (kitörések, napfoltok stb.)(V, variability), a műszer által előidézett zaj (IS, instrumental noise), vagy egyéb azonosíthatatlan esemény következett be (J, junk, O, other). Emiatt már alkalmas lehet az adathalmaz gépi tanulásra, hiszen adottak az osztálycímkék. Ezek a címkék a “Disposition” oszlopban találhatók. Fontos megjegyezni, hogy a felcímkézés nem hibátlan, felléphetnek hibásan osztályozott példák is. A fájlról a publikáció 3. oldalán írnak részletesebben. Gépi tanuláskor a V, IS, J és O címkéket egységesen J címkére cserélték, ugyanis ezek megkülönböztetése jelen esetben érdektelen. Vizsgáljuk meg, hogy milyen az így kapott három osztálycímkénk eloszlása! Az adatok 83,5%-a J, 13,4%-a EB és csupán 3,0%-a tartozik a PC osztályba. A 6. ábrán látható, hogy az adathalmaz erősen kiegyensúlyozatlan.



***6. ábra**. Az osztályok eloszlása a tces.csv fájlban. Saját készítésű ábra.*

A tces.csv fájl minden sorához elkészítették a globális, lokális és másodlagos nézeteket. A globális nézethez 201, a lokális és másodlagos nézetekhez 61-61 bin értéket választottak, a QLP pipeline használatával. A fénygörbéket ezután a BLS (Box Least Squares algoritmus; Kovács et al., 2002) által meghatározott periódusba hajtogatták, így a tranzitokat egy vonalba és középre helyezték. Eltávolították a nullától eltérő adatminőség-jelzőkkel rendelkező képeknek megfelelő pontokat, és minden olyan felfelé kiugró értéket, amely több mint ötszöröse a medián abszolút eltérésének a mediántól. Ezen kívül egy “depth change” értéket is felvettek minden méréshez, két különböző méretű. 2,75 és 3 pixeles sugárral rendelkező nyílásban mért tranzitmélységek különbségét elosztották a kisebb nyílásban mért tranziton kívüli szórásértékkel. A tranzimélységek becsléséhez egy dobozmodellt illesztettek a fénygörbére. Ezt a „mélységváltozás” jellemzőt úgy normalizálták, hogy kivonták az egész tanítókészlet átlagát, és elosztották a szórásértékkel. Végül az adathalmazt 80% tanító, 10% validáló és 10% tesztelő halmazra bontották. Két bináris osztályozási feladatot határoztak meg, a “triage” esetében az EB és PC címkével rendelkező osztályokat próbálják meg elválasztani a J címkéjű osztálytól, “vetting”-nél pedig a PC osztályt a többitől. A triage alkalmas arra, hogy kiszűrjék a rengeteg érdektelen adatot, megkönnyítve ezzel a csillagászok munkáját, ugyanis így egy csökkentett adatmennyiséget kell csak kézzel, egyesével átvizsgálniuk. Ezen feladat során magas eredményeket értek el, 97%-os átlagos precision és 97%-os accuracy értéket produkáltak, így a modell már éles alkalmazásra is alkalmas, a csillagászok azóta használják is. A feladathoz Shallue & Vanderburg (2018) hálóját alkalmazták, és ehhez igazodva csupán a globális és lokális nézeteket használták inputként. Bár az eredmények biztatóak, ahogy látható volt, fedési kettőscsillag-rendszerekből sokkal többet detektál az űrtávcső (több, mint négyszer annyit, mint exobolygó jelöltet), így a kapott eredménynek még mindig csak egy kis része exobolygó valójában. Ráadásul 100%-os recall értéket adó threshold mellett, vagyis amikor minden EB-t és PC-t megtalált a háló, a precision érték körülbelül 60%, vagyis 10-11% körüli része exobolygó csak a predikált halmaznak. A vetting feladat ennél jóval nehezebb, az elért eredmények is ezt tükrözik: 97,8% accuracy mellett 69,3%-os átlagos precision értéket értek el. A vetting feladat során használt neurális hálózatuk szerkezete a 7. ábrán látható.



7. ábra

Az AstroNet-Vetting modell.

Liang Yu és munkatársai által használt neurális hálózat modellje.

Forrás: Yu, L. et al. (2019), p. 6.

6. Triage

Korábban említettem, hogy ezen kutatási területen két külön feladat definiálása általános, egy “triage” és egy “vetting” feladatot (pl. Yu, L. et al., 2019; Malik, A. et al., 2022). A dolgozatban a triage feladatra nem fektettem nagy hangsúlyt, kifejezetten a vetting, vagyis a bolygójelöltek megtalálása volt a célom.

Bár a vetting fejlesztése az irányadó elv a legtöbb exobolygó detektáló publikációban, valójában a két feladat szorosan kapcsolódhatna egymáshoz. Egy hierarchikus modellel első lépésben kiszűrhetnénk a J osztályba tartozó elemeket (triage), majd utána a megmaradt adathalmazon végezhetnénk el a “vetting” feladatot. Sajnos a jelenlegi triage feladatra készült modellek teljesítménye még nem olyan magas, hogy ezzel jobb eredményeket érjek el.

A dolgozat további részében a vettingről írok, a triaget csupán a 7.3.1-es pontban említem érintőlegesen.

7. Vetting

7.1. Célkitűzés

Az adathalmaz erős kiegyensúlyozatlansága miatt az accuracy érték nem túl mérvadó, nem hordoz túl sok információt. Az adatok csupán 3%-a tartozik a PC osztályba, így ha írunk egy olyan algoritmust, amely minden inputot a “J” osztályba sorol, akkor az accuracy értékünk 97% lesz. Ilyen magas accuracy érték azt sugallhatja, hogy a hálónk teljesítménye kiváló, azonban a valóságban semmit sem tanult meg. Emiatt a precision és recall értékeket tekinthetjük kulcsfontosságú metrikáknak. Azon belül is a magas recall érték elérése a célom, vagyis a precision-recall görbén a magas (90-95%) recall értékek precision-jét szeretném fejleszteni. Úgy gondolom, hogy a legfontosabb, hogy a lehető legkevesebb exobolygót veszítsük el a predikálás során, egy magas recall és alacsony precision értékű modellt értékesebbnek tekintek egy magas precision és alacsony recall értékű modellnél.

7.2. Az adatok parszolása

Az MIT-s adathalmazt tensorflow recordokként bináris formában töltötték fel, így először is ezeket könnyebben kezelhetővé konvertáltam. Írtam egy függvényt, amelynek segítségével a három adathalmazt pandas dataframekké alakítottam. A tanítás során a “tic_id”, a globális, lokális, másodlagos nézeteket és a “depth_change” értékeket használtam. A parszolás során az adatok nem sérülnek, az értékek nem módosulnak semmilyen mértékben.

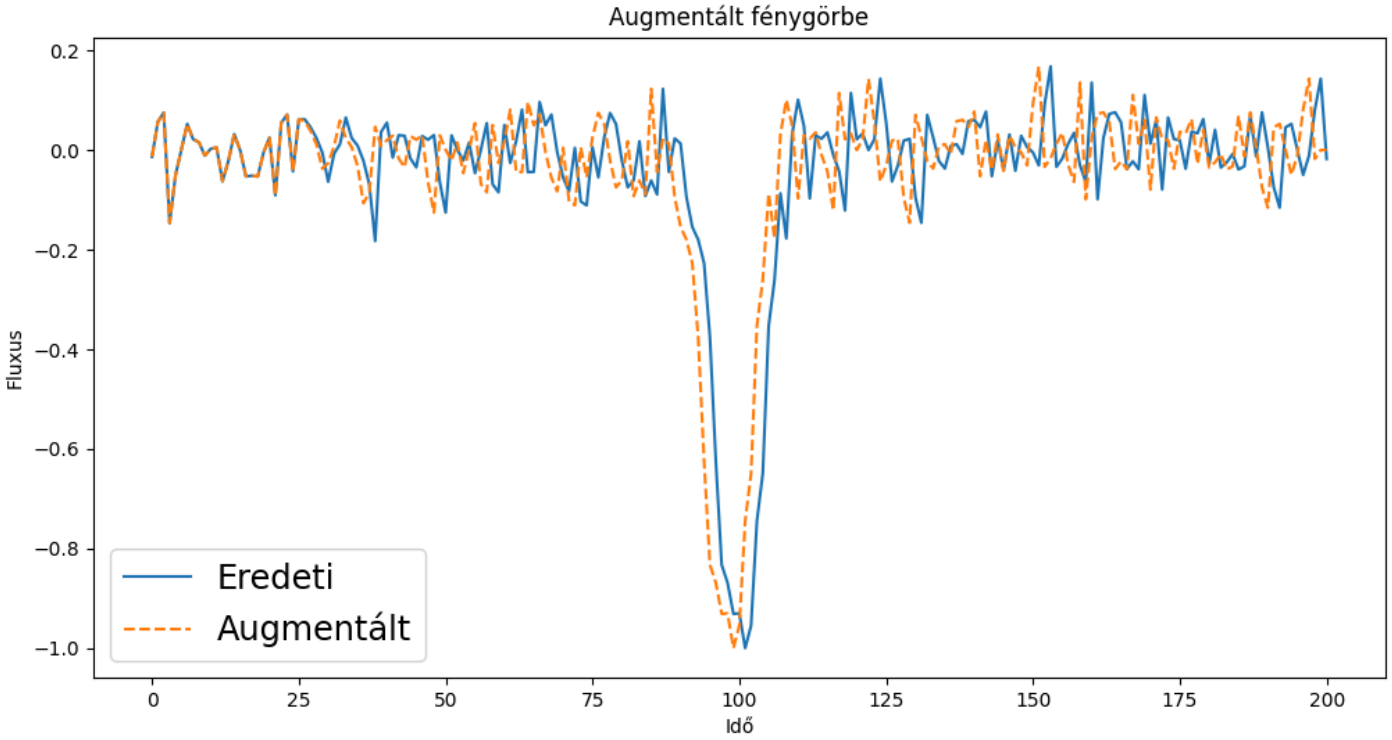
7.3. Kiegyensúlyozatlanság kezelése

7.3.1. SMOTE

Az osztályok erős kiegyensúlyozatlansága aggasztó, és kezdetben nem sikerült értékelhető eredményeket elérnem emiatt. A kiegyensúlyozatlanságot orvosolhatnánk szintetikus túlmintavételezéssel, például a SMOTE (Synthetic Minority Over-sampling Technique) vagy a Borderline-SMOTE (a legnehezebben elkülöníthető adatokra fókuszáló verziója a SMOTE-nak) segítségével. (Han et al., 2005) A SMOTE először kiválaszt egy mintapéldányt a kisebbségi osztályból, majd megkeresi a szomszédokat és ezután megtalálja ennek a mintának a legközelebbi szomszédait, általában az euklideszi távolság alapján. Majd választ egy szomszédot véletlenszerűen, és az eredeti minta és a kiválasztott szomszéd közötti vektoron egy véletlenszerűen kiválasztott ponton létrehoz egy új, szintetikus mintapéldányt. Ezáltal a kisebbségi osztály mintái közötti interpolációval új példákat generál. Sajnos a technika nem tudja megfelelően modellezni az időbeli jellemzőket, egyszerű interpolációt alkalmaz, amely erősen torzíthatja a modell által tanult mintázatot. A triage feladatnál minimálisan jobb eredményt értem el a Borderline-SMOTE technikát alkalmazva, hiszen ott egyszerűbb a két osztály elválasztása és a fellépő torzítások nem olyan számottevőek. Azonban a SMOTE és a Borderline-SMOTE a vetting feladat során hátráltatják a modellt, minden esetben rosszabb eredményeket ért el ilyen fajta túlmintavételezéssel, mint anélkül.

7.3.2. Adataugmentáció

Kiegyensúlyozatlan halmaz esetén hagyományos megoldás lehet még az adataugmentáció, viszont kellő körültekintéssel szabad csak ilyen eljárást használnunk, mivel teljesen tönkretehetik az adatainkat. Idősori elemzéseknél például klasszikus augmentációnak számít a függőleges tükrözés, vagy a görbe felszeletelése és az egyes részek felcserélése. Ezek olyan torzítások, amelyek nem növelik az exobolygó fénygörbéinek a diverzitását, csupán további értelmetlen adatokat generálnak. Ha egy fénygörbét feldarabolunk és felcseréljük a szeleteket, akkor elveszítjük a struktúrális információját. A Yu, L. et al. (2019) publikációban véletlenszerű vízszintes tükrözéseket használtak 0.5-ös valószínűség mellett, ezt én is implementáltam. Ezen felül készítettem egy "warp_light_curve" függvényt, amely segítségével a görbe egy szeletét kis mértékben elnyújthatjuk/összenyomhatjuk vízszintes irányba (8. ábra).. A módszer hasonlít ahhoz, ahogyan a lokális nézetet készítjük, viszont itt nem arányosan az egészet nagyítom, csupán egy véletlenszerű részt. A torzítás mértékét a "warp_factor" paraméter segítségével tudjuk beállítani. A tanításaim során 0.08-as értéket használtam. Ezzel a technikával növelni tudtam a hálóm teljesítményét.



8. ábra *A "warp_light_curve" eljárásommal készített adataugmentáció. Saját szerkesztésű ábra.*

7.3.3. Adatok szűrése a paraméterek és metaparaméterek hangolásához

Az adathalmaz sok olyan adatot tartalmaz, amelyek könnyen szeparálhatóak. A J osztály számos olyan példát tartalmaz, amelyek egyértelműen nem exobolygók. A háló minden tanításakor folyamatosan végignézi ezeket az adatokat, amelyeket könnyedén osztályozna, viszont így megzavarhatják a tanulásban, és kevésbé tud fókuszálni a nagyon nehezen szeparálható osztályok komplex elkülönítésére. Ezen felül az adataugmentáció során ezeket az adatokat is augmentáljuk, vagyis még több ezekhez hasonló fölösleges adatot generálunk, amelyek ráadásul drasztikusan növelik a kód futásidejét. A megnőtt tanítási idő pedig kevesebb lehetőséget ad arra, hogy finomhangoljuk a háló paramétereit, metaparamétereit. Ennek érdekében a tanítás előtt egy egyszerű szűréssel eltávolítom az adatok egy részét, így egy csökkentett adathalmazt kapok, amely segítségével gyorsabban elvégezhető a finomhangolás. Végül majd a teljes adathalmazon tanítom és értékelem ki a hálót. A tanításaim során nem volt példa olyan esetre, amikor egy változtatás után a szűrt adathalmazon a modell jobb teljesítményt ért volna el, de a teljes adathalmazon rosszabbat, vagy fordítva.

Vannak általános elvárásaim az exobolygók fénygörbéjével kapcsolatban. Ezek az elvárásaim a következők: (1) a globális és lokális nézeteknél középen legyen egy kitűnő fénycsökkenés, (2) az előtte-utána lévő értékeknél pedig lehetőség szerint ne lépjen fel ilyen. Ennél bonyolultabb elvárásokat nem szeretnék szabni, mivel a bemutatott ábrákon látszik, hogy a minta olyan zajos, hogy szigorúbb feltételek mellett értékes adatokat veszíthetünk. A két küszöbértéket, azaz, hogy milyen mértékben legyen középen átlagosan fénycsökkenés, és legfeljebb milyen mértékben a két szélén, a tanító adathalmaz alapján határozom meg. Fontos, hogy a tesztelő adathalmazt semmilyen augmentáció vagy szűrés során sem vizsgálhatjuk (peeking történne). Ha arra szabjuk az értékeinket, akkor hamis eredményeket fogunk kapni. A háló robosztusságáról, általánosító képességéről nem szerzünk információt, csupán a teszthalmazra optimalizáltuk azt.

A lokális nézetnél ha az első és utolsó 20 pont átlaga nem haladja meg a 0.18-as értéket, a középső 21 pont átlaga pedig -0.3-nál kisebb, akkor megtartjuk, másodlagos nézet esetében pedig elvetjük. Globális nézet esetében akkor tartjuk meg az adatot, ha az első és utolsó 98 pont átlaga 0.05-nél kisebb, a középső 5 pont átlaga pedig -0.2-nél kisebb. Ezek a határok nagyon megengedőek, úgy állítottam be őket, hogy egyáltalán ne, vagy rendkívül minimális mértékben veszítsek exobolygó jelölt példákat. A modell első futtatása után

kiderült, hogy a teszthalmazból 1 exobolygójelöltet sajnos elveszítettem a szűréskor, ám az a sejtésem, hogy amelyik fénygörbe még ezt a minimális kritériumot sem teljesítette, annak a PC osztályba való sorolása eleve közel lehetetlen feladat lett volna. Viszont a teljes adathalmaz majdnem a negyedére csökkent, és a futási idő is ezzel arányosan csökkent.

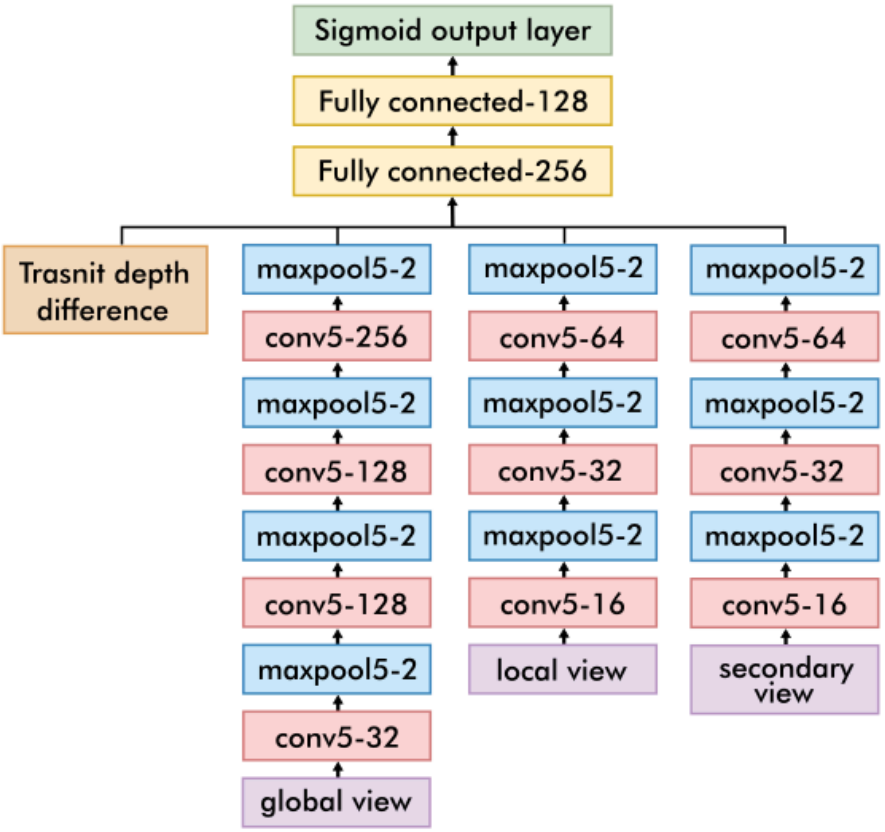
7.3.4. Kiegyensúlyozott batch generálás

Kiegyensúlyozatlan halmazon történő tanítás javítható kiegyensúlyozott batch generálással (Chawla et al., 2004; He, H., & Garcia et al., 2009). A módszer egyenletesen elosztott mintákat szolgáltat a különböző osztályokból minden egyes batch esetében. Ezáltal a modell egyenlő mértékben tanulhatja meg az adatokat minden osztályból, ami csökkentheti a túltanulás esélyét azoknál az osztályoknál, amelyek túlreprezentáltak az adathalmazban (Osborn et al., 2019). A tanítás során az "inbalanced-learn" Python csomag "BalancedBatchGenerator" nevű osztályát használom. A batch-ek kiegyensúlyozásával jobb eredményeket értem el, mint például a veszteségfüggvény súlyozásával, ami szintén egy elterjedt módszer kiegyensúlyozatlan adathalmazok esetén. BalancedBatchGenerator: https://imbalanced-learn.org/stable/references/generated/imblearn.keras.BalancedBatchGenerator.html

7.4. A neurális hálózat

A neurális hálózatom szerkezete eltérő az AstroNet-Vetting modelltől. Kevesebb réteget és kevesebb neuront használtam. Két CNN ágat készítettem, az egyiket a globális nézet feldolgozásához, a másikat a lokális és másodlagos nézetekhez. Minden konvolúciós réteg után batch normalizálást alkalmaztam, valamint relu aktivációt, 1D-s max pooling-ot, majd dropout-ot a robosztusság növelése érdekében. A dropout érték a mélyebb ág esetében 0,3; 0,4; 0,4; 0,5 a sekélyebb ágnál pedig 0,2; 0,3; 0,4 vagyis a mélyebb rétegek felé növekedtek. L2-es regularizációt alkalmaztam minden konvolúciós rétegnél l2 = 0.001 értékkel. Az ágakat a teljesen összekötött rétegeknél egyesítettem, amely csupán 2 rétegből áll, egy 256 és egy 128 neuron szélességűből, 0.4 majd 0.5-ös dropout értékekkel. A teljesen összekötött rétegeknél vettem számításba a mélységcsökkenés paramétert is, hasonlóan az AstroNet-Vetting modellhez. Az adathalmazban szereplő "depth_change" értékekből a

kiugró értékeket kiszűrtem, hogy majd a “tranzit mélység különbség” értékeit ne tegye túl homogénné a kiugró értékek torzító ereje az átlag és a szórás számításakor, ugyanis az adathalmaz tartalmaz ilyen rendkívül markáns értékeket. A legalsó és legfelső 1%-ot tekintettem outlier értékeknek. Ha egy adatérték kisebb, mint az alsó korlát, akkor az alsó korlát értékére, ha nagyobb, mint a felső korlát, akkor a felső korlát értékére módosul. Az így kapott értékek elég kicsik lettek, ezért négyszeres szorzóval felnagyítottam őket, máskülönben gyakorlatilag nem lenne szerepünk a predikáláskor. Viszont túl dominánsak sem lehetnek. (9. ábra)



9. ábra

A saját neurális háló szerkezetem.

Saját szerkesztésű ábra.

7.5. A modell tanítása

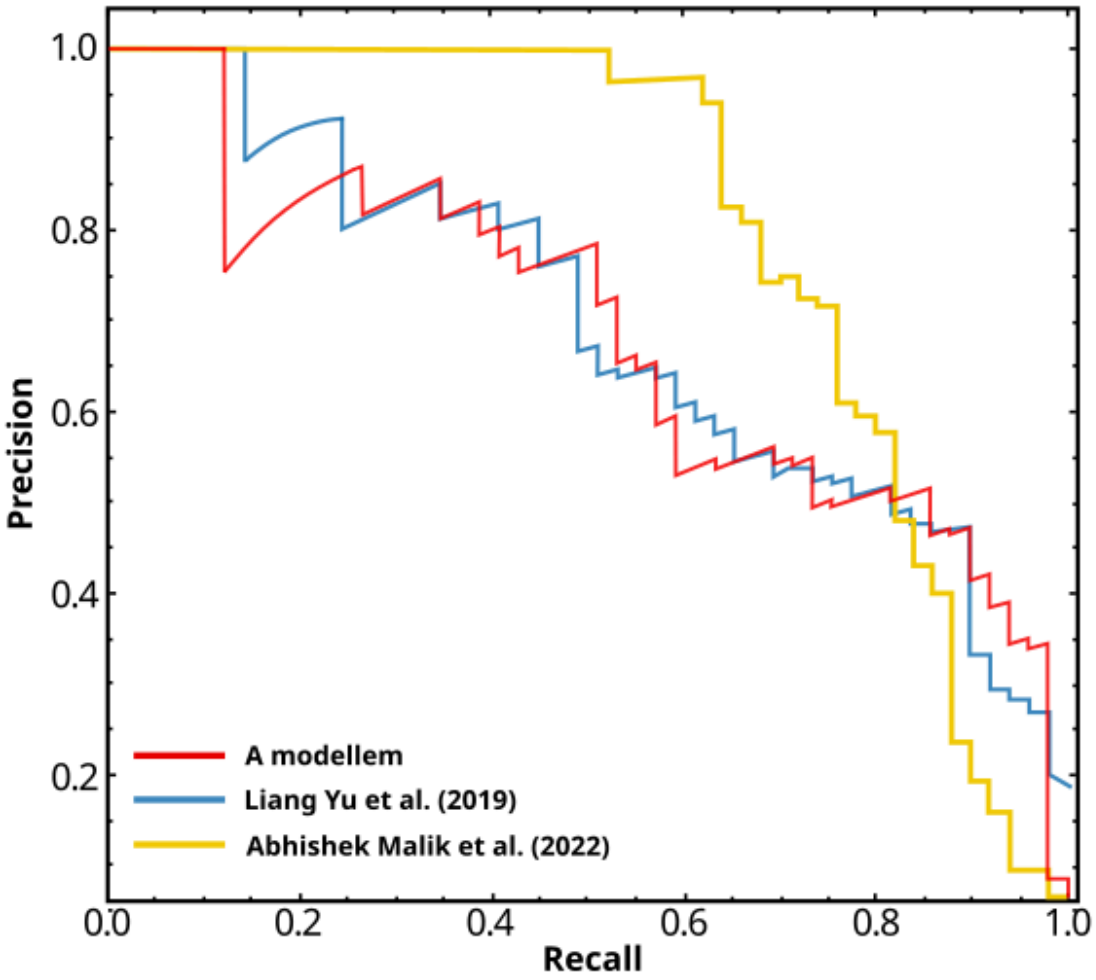
A modell tanítása során Adam optimalizálót használtam, veszteségfüggvénynek pedig kategorikus kereszt-entrópiát. A tanulási ráta beállításához ütemezőt alkalmaztam a "LearningRateScheduler" segítségével. Az első két epochig nem változtatok a rátán (kezdetben 0.001-es rátát használok), majd utána e^(-0.1) értékkel szorzom. A túltanulás elkerülésének érdekében korai megállást állítok be, ahol a validációs halmaz veszteségét figyelem, 10-es patience érték mellett, vagyis mindig, amikor a veszteség értéke csökken, elmentjük a modellt, majd ha 10 epochon keresztül nem történik újabb csökkenés a legalacsonyabb értékünkhöz képest, a tanítás megáll, és a legutoljára elmentett modellel értékeljük ki a teszthalmazt (Bai et al., 2021). A modellt 30 epochig tanítottam 75-ös batch mérettel.

A modellem egyszeri futtatása nem tükrözi az általános teljesítményét, olykor véletlenül elérhet kiugróan magas teljesítményt a teszthalmazon, így Liang Yu és munkatársai módszerével megegyezően egymás után tízszer futtattam (a súlyok véletlen inicializálódnak), és az eredményeket átlagoltam, hogy egy átfogóbb képek kapjak. Ennek a megközelítésnek az esetleges problémáira a dolgozat 8.2. pontján térek ki. A precision-recall görbe kirajzolásával pedig áttekinthetjük, hogy a threshold érték változtatásával hogyan változik a precision és recall érték.

7.6. A modell kiértékelése

A modell 97,5%-os recall érték mellett 36,0%-os precision értéket ért el. Liang Yu és munkatársai (2019) 97,5%-os recall érték mellett 27% körüli precisiont értek el, Abhishek Malik és munkatársai (2022) pedig 10% körüli precisiont. A 10. ábrán látható, hogy az Abhishek-féle megközelítés nagyon magas precisiont eredményez alacsony recall értékek mellett, azonban így elveszítik az exobolygók jelentős részét. A magas recall értékek esetében pedig jóval alacsonyabban teljesít a modellemnél. Liang Yu és munkatársai 89,8%-os recall mellett 57%-os precisiont értek el, azonban kicsit magasabb recall értékre már drasztikusan esik a hálójuk teljesítménye. Az én modellem szintén 57%-os precisiont ér el 90% körüli recall érték mellett, azonban nem történik szignifikáns zuhanás a

teljesítményében 97,5%-os recall-ig. A recall tengelyen 0.85-től 0.975-ig tartó tartományban végig magasabb precision értékeket ér el a modellem Malik, A. et al. (2022) és Yu, L. et al. (2019) eredményeinél.



10. ábra

A precision-recall görbe.

Yu, L. et al. (2019) és Malik, A. et al.(2022) alapján készített saját szerkesztésű ábra.

8. Kritikai észrevételek Yu, L. et al. (2019) eredményeivel kapcsolatban

8.1. Adatszivárgás

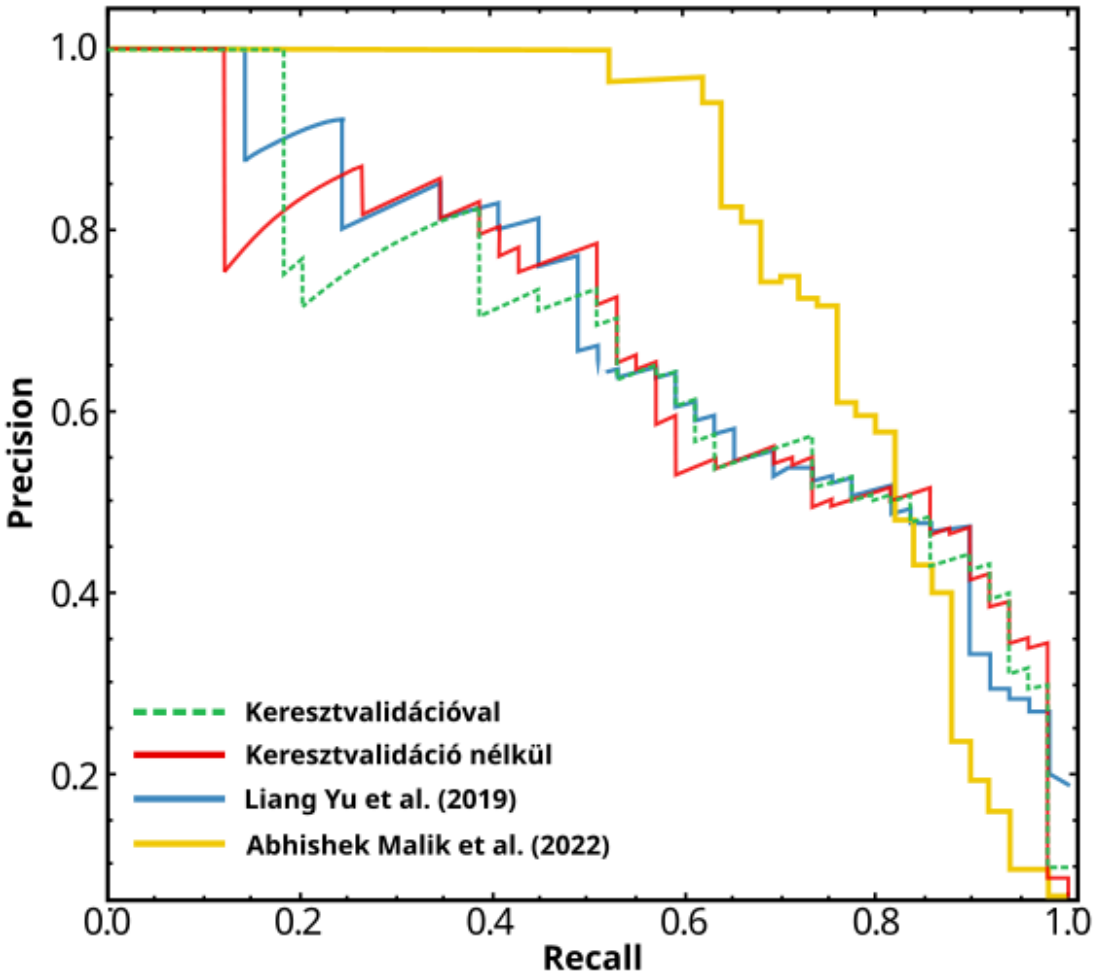
A TESS adathalmazon történő kezdeti vizsgálataim során a már említett tces.csv fájl alapján én kértem le a fénygörbe adatokat és készítettem el belőlük a megfelelő nézeteket. A SPOC pipeline-t használtam, különböző bin értékekkel kísérleteztem, hogy vajon hogyan lehetne a lehető legjobban reprezentálni a fénygörbéket. Savitzky–Golay filterrel simítottam, [0,1] intervallumra normalizáltam az értékeket, megkíséreltem az adatok legoptimálisabb előkészítését. Azonban a saját adathalmazomon nem sikerült megközelítenem a Liang Yu et al. 2019 publikációban közölt accuracy, precision, recall értékeket. Megpróbáltam magyarázatot találni arra, hogy miért eredményesebb az ő megközelítésük. Az adatok vizsgálata során észrevettem, hogy a teszthalmazban megtalálhatóak olyan adatok, amelyek jelen vannak a tanító adathalmazban is. (Megjegyzem, hogy a saját adathalmazomon elért gyengébb eredmény mértéke nem arányos az átfedéssel, Liang Yu és munkatársainak előfeldolgozása valóban kifinomultabb és eredményesebb megközelítés.) Úgy gondolom, hogy ez “data leakage”-nek, vagyis adatszivárgásnak minősül. Azokat a példákat tanítás során már látta a háló, így könnyen ráismer azokra a tesztelés során, azonban ez nem ad valós képet a háló általánosító képességéről. A publikációjukban megemlítették, hogy a tces.csv fájl esetleg tartalmazhat pár duplikációt, azonban szerintük ezeknek nem lehet számottevően befolyásoló hatása. A tces.csv fájl tartalmaz duplikált TIC ID-kat, azonban egy részénél eltérő a többi érték, más szektorokban figyelték meg az adott objektumot (valószínűleg a szektorok közötti átfedés miatt) és más a becsült periódus, tranzit időtartama és további paraméterei. Erre tekinthetünk egyfajta adataugmentációként is, mivel lényegében egy objektumhoz tartozó mérés van kis mértékben módosítva. Azonban nem ezek az esetek torzítják az eredményeket, hanem az olyan duplikációk, amelyek értéke minden elemében megegyező. Összesen 100 olyan eset van, amikor a teszthalmazban egy sor értékei pontosan megegyeznek a tanítóhalmazban lévőkkel, vagyis amikor a globális, lokális és másodlagos nézetek is az utolsó értékig tökéletesen egyeznek. Az adathalmaz kellően nagy, így teljesen értelmetlennek látom ezt az átfedést, ezeket az eseteket még a legelején érdemes lett volna kiszűrniük, ugyanis nem vesztettek volna sok adatot, viszont nem merült volna fel az

adatszivárgás esetleges befolyásoló hatása az eredményeikre. A tesztelő és tanító adathalmazok közötti átfedést nem említik a publikációban. A többi általam ismert publikációban, amelyek az ő adataikat használják sem említik az adatszivárgást. Az MIT által kiadott adathalmaz szerkezetére, adataira nem térnek ki, csak a kész, megadott tensorflow record fájlokat használják.

8.2. Robosztusság, keresztvalidáció

Liang Yu és munkatársai tíz véletlenszerű inicializációval indították a modelljeik tanítását, majd a 10 modell predikcióit átlagolták, hogy fejlesszék a neurális hálózatuk robosztusságát. Azonban ezt a tíz tanítást ugyanazzal a validációshalmazzal és tanítóhalmazzal végezték. Korábban megmutattam, hogy az adathalmaz egyes példái olykor nagyon nehezen osztályozhatók és az adathalmaz erősen kiegyensúlyozatlan. Emiatt a modell teljesítményét a teszthalmazon erősen befolyásolhatja, hogy mely elemek kerültek a validációshalmazba. Ennek szemléltetéseként a tanító- és validációshalmazt összeolvasztottam, az elemeket “összekevertem” és véletlenszerűen újraosztottam. Öt ilyen újraosztást végeztem el, majd ezeken a halmazokon tanítottam a neurális hálózatomat Liang Yu és munkatársai elve szerint. Tíz véletlenszerű inicializációval tanítottam tíz modellt minden ilyen újraosztásra, majd megnéztem, hogy az öt újraosztás során hogyan teljesítettek a modellek. Az egyik tanítóhalmaz-validációshalmaz páros jóval magasabb teljesítményt eredményezett. Jelen esetben ha a modell robosztusságáról szeretnénk képet kapni, akkor érdemesebb keresztvalidációt végeznünk, és ezeket az eredményeket átlagolni, nem pedig egy fix validációshalmazra elvégezni több véletlenül inicializált tanítást.

Az erős kiegyensúlyozatlanság miatt 5 részre osztottam az egyesített halmazt, hogy egy adott halmazba lehetőség szerint minél több PC osztálybeli egyed essen. A 11. ábrán látható a modellem precision-recall görbéje ötszörös keresztvalidáció után. A modellem teljesítménye kis mértékben csökkent, azonban a korábban említett magas recall értékek mellett még mindig szép eredményeket ért el. Jól szemlélteti az ábra, hogy a keresztvalidációval eshet a teljesítmény. Ezzel szemben a háló robosztussága megnő. Liang Yu és munkatársai nem végeztek keresztvalidációt, így a hálójuk robosztusságáról valójában nem kaptunk pontos képet. A kísérleteim azt sejtetik, hogy a publikált eredményeiknél ők is kisebb értékeket értek volna el ilyen kiértékeléssel.



11. ábra

A precision-recall görbe kiegészítve a keresztvalidációval tanított modellem eredményeivel.

Yu, L. et al. (2019) és Malik, A. et al.(2022) alapján készített saját szerkesztésű ábra.

9. TESS összegzés

A dolgozatban érintettem olyan témákat, amelyekkel növelhető az exobolygó detektáló modellek teljesítménye. Ezen felül megmutattam, hogy kevesebb neuronszámú, sekélyebb hálóval is elérhető a Yu, L. et al. (2019) által bemutatott teljesítmény, sőt a számomra kritikusnak tekintett magas recall értékek esetében túlnyomó részben kiemelkedőbb precision értékek elérése is.

A kiegyensúlyozott batch normalizálás és az adataugmentációk segítették a tanítást, mérsékelték az erős kiegyensúlyozatlanság miatt fellépő problémákat. Az általam bemutatott adataugmentáció széles körben alkalmazható egyéb idősori adatoknál is számos területen.

A mélytanulás egyik hátránya, hogy a paraméterek, metaparaméterek finomhangolása rengeteg időt vehet igénybe. Erre bemutattam egy egyszerű szűrőalgoritmust, amelynek segítségével az adathalmaz mérete csökkenthető. A csökkentett adathalmaz megőrzi a nehezen osztályozható példák túlnyomó többségét, és ezáltal a háló gyorsabban optimalizálható.

Végül felvetettem két problémát Yu, L. et al. (2019) módszereivel és a modellük robosztusságával kapcsolatban.

10. További irányok és kitekintés

A csillagok egyéb jellemzőinek felhasználásával tovább fejleszthető a modellek teljesítménye. Sajnos a TESS adatok esetében jelenleg ezen értékek erősen hiányosak, ezért a dolgozatban ilyen jellemzőket nem használtam, hogy minél több mérést feldolgozhassak.

A kutatások során egyre népszerűbb irány a “centroid”-ok, vagyis a súlypontok felhasználása. Osborn et al. (2020) megmutatta, hogy (a dolgozatban bemutatott adatoktól eltérő szektorokban készült adatok esetében) a centroid görbék felhasználása növelheti a modellek teljesítményét. A lightkurve modul segítségével (Cardoso, J. et al., 2018) sajnos rengeteg mérés esetén nem tudtam lekérni ezen adatokat, vagy az adatok csupán 30%-a állt rendelkezésre, így a dolgozatban nem használtam ilyen adatokat. Az adathalmazom a töredékére csökkent volna, így nezehen összehasonlítható eredményt értem volna el.

A dolgozatban bemutatott módszerek alkalmasak lehetnek kettőnél több osztályos predikciókra is, egyes kutatások kifejezetten a fedési kettőscsillag-rendszerek vizsgálatára épülnek, így ezt a három osztályt akár egy modellel is szétválaszthatjuk. Ez a megközelítés mindenképpen igényli a háló további hangolását, illetve a szűrőalgoritmus korlátainak igazítását.

A TESS adatok mennyisége folyamatosan nő, az adathalmaz növekedése pedig rendkívül kedvező lehet a neurális hálók eredményeinek fejlesztése szempontjából (Tey, E. et al., 2023). A dolgozatban csupán az első öt szektor adatait használtam, azonban a többi szektor adatainak felhasználásával tovább növelhető a modell robosztussága. Már az első öt szektor adatainak esetében is kedvezőnek bizonyult egy szűrőalgoritmus bevezetése a háló finomhangolásához (a korlátozott számítási kapacitás miatt), úgy gondolom egy nagyobb adathalmaznál különösen hasznos lehet egy szűrőalgoritmus használata.

A dolgozat az exobolygók detektálására fókuszált, azonban a TESS fejezetben leginkább idősori adatokat elemeztem. Ezen kutatási területen elért eredmények fejlesztései felhasználhatók lehetnek egyéb periodikus idősori jelenségek vizsgálatára is (pl. orvosi szakterületen EKG (elektrokardiográfia) mérések feldolgozására, pénzügyi területen periodikusan bekövetkező gazdasági trendek vizsgálatára).

Irodalomjegyzék

Bai, Y., Yang, E., Han, B., Yang, Y., Li, J., Mao, Y., ... & Liu, T. (2021). Understanding and improving early stopping for learning with noisy labels. Advances in Neural Information Processing Systems, 34, 24392-24403.

Cardoso, J. V. D. M., Hedges, C., Gully-Santiago, M., Saunders, N., Cody, A. M., Barclay, T., ... & Lightkurve Collaboration. (2018). Lightkurve: Kepler and TESS time series analysis in Python. Astrophysics Source Code Library, ascl-1812.

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Special issue on learning from imbalanced data sets. ACM SIGKDD explorations newsletter, 6(1), 1-6.

Coughlin, J. L., Mullally, F., Thompson, S. E., Rowe, J. F., Burke, C. J., Latham, D. W., ... & Zamudio, K. A. (2016). Planetary candidates observed by Kepler. VII. The first fully uniform catalog based on the entire 48-month data set (Q1–Q17 DR24). The Astrophysical Journal Supplement Series, 224(1), 12.

Han, H., Wang, W. Y., & Mao, B. H. (2005, August). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In International conference on intelligent computing (pp. 878-887). Berlin, Heidelberg: Springer Berlin Heidelberg.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9), 1263-1284.

Kovács, G., Zucker, S., & Mazeh, T. (2002). A box-fitting algorithm in the search for periodic transits. Astronomy & Astrophysics, 391(1), 369-377.

Kramer, O. (2013). K-Nearest Neighbors. In: Dimensionality Reduction with Unsupervised Nearest Neighbors. Intelligent Systems Reference Library, vol 51. Springer, Berlin, Heidelberg.

Kramer, O., & Kramer, O. (2016). Scikit-learn. Machine learning for evolution strategies, 45-53.

Lowell Observatory. (n.d.). History of Pluto. Retrieved from https://lowell.edu/discover/history-of-pluto/

Malik, A., Moster, B. P., & Obermeier, C. (2022). Exoplanet detection using machine learning. Monthly Notices of the Royal Astronomical Society, 513(4), 5505-5516.

McCauliff, S. D., Jenkins, J. M., Catanzarite, J., Burke, C. J., Coughlin, J. L., Twicken, J. D., ... & Cote, M. (2015). Automatic classification of Kepler planetary transit candidates. The Astrophysical Journal, 806(1), 6.

Montenbruck, O., & Gill, E. (2000). Satellite Orbits: Models, Methods, and Applications. Springer Science & Business Media.

NASA's High Energy Astrophysics Science Archive Research Center. (n.d.). Eclipse. Retrieved from https://heasarc.gsfc.nasa.gov/docs/RXTE_Live/eclipse.html

NASA's Imagine the Universe. (n.d.). More about light curves. Retrieved from https://imagine.gsfc.nasa.gov/features/yba/M31_velocity/lightcurve/lightcurve_more.html

NASA's Imagine the Universe. (n.d.). Timing. Retrieved from https://imagine.gsfc.nasa.gov/science/toolbox/timing2.html

NASA. (n.d.). Kepler: A search for habitable planets. Retrieved from https://science.nasa.gov/mission/kepler/

NASA. (n.d.). TESS: Transiting Exoplanet Survey Satellite. Retrieved from https://science.nasa.gov/mission/tess/

NASA. (n.d.). What's a transit? Retrieved from https://science.nasa.gov/exoplanets/whats-a-transit/

Osborn, H. P., Ansdell, M., Ioannou, Y., Sasdelli, M., Angerhausen, D., Caldwell, D., ... & Smith, J. C. (2020). Rapid classification of TESS planet candidates with convolutional neural networks. Astronomy & Astrophysics, 633, A53.

Peterson, L. E. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.

P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," in IEEE Transactions on Geoscience Electronics, vol. 15, no. 3, pp. 142-147, July 1977, doi: 10.1109/TGE.1977.6498972

Ruczinski, I., Kooperberg, C., & LeBlanc, M. (2003). Logic Regression. Journal of Computational and Graphical Statistics, 12(3), 475–511

Shallue, C. J., & Vanderburg, A. (2018). Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. The Astronomical Journal, 155(2), 94. DOI 10.3847/1538-3881/aa9e09

Soop, E. M. (1994). Handbook of Geostationary Orbits. Springer.

Tey, E., Moldovan, D., Kunimoto, M., Huang, C. X., Shporer, A., Daylan, T., ... & Seager, S. (2023). Identifying Exoplanets with Deep Learning. V. Improved Light-curve Classification for TESS Full-frame Image Observations. The Astronomical Journal, 165(3), 95.

Yu, L., Vanderburg, A., Huang, C., Shallue, C. J., Crossfield, I. J., Gaudi, B. S., ... & Quinn, S. N. (2019). Identifying exoplanets with deep learning. III. Automated triage and vetting of TESS candidates. The Astronomical Journal, 158(1), 25. DOI 10.3847/1538-3881/ab21d6

Appendix

1. A dolgozathoz felhasznált mesterséges intelligencia alapú eszközök

A dolgozat során a csatolt kódok debugolásához, illetve a precision-recall görbe kirajzolásához szükséges kód generálásához a ChatGPT mesterséges intelligencia alapú eszközt használtam.

2. A dolgozat során használt Colab füzeteim és modelljeim

A dolgozat során használt Colab füzeteim és modelljeim elérhetősége:

https://github.com/gergotorok02/Exobolygo_detektalas-1