Mélytanulás projekt beszámoló

Csapatnév: Pain and Panic

Csapattagok: Tugyi Beatrix (T63K63), Heizer Levente (IT9P0)

1. Bevezetés

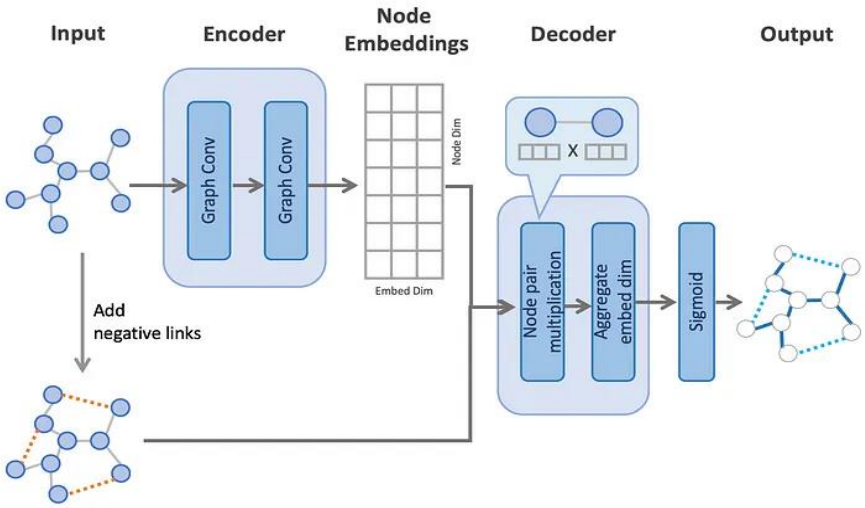
A gráf neurális hálók (GNN) a mesterséges intelligencia és mélytanulás területén jelentős eredményeket értek el az elmúlt években. [1],[2] A GNN-ek a hagyományos neurális hálók módszertanára épülnek és kifejezetten jól alkalmazhatók számos tudományos területen komplex struktúrájú adathalmazok feldolgozására. [3]

Ez alól a társadalomtudományok sem alkotnak kivételt, ahol az emberi viselkedést mellett a társadalmi és kulturális jelenségeket vizsgálják. Az emberi viselkedés elemzése során kiemelten fontos az emberi kapcsolatok és interakciók megértése. Ez történhet ún. ego hálózatok elemzésével is, ami segít megérteni egy ego, azaz egy konkrét személy, kapcsolati interakcióit és annak dinamikáját. [4]

2. Elméleti háttér

Az ego hálózatok elemzésére egy hatékony és erős eszköz a gráf neurális hálózatok (GNN) alkalmazása. Ezekben a hálózatokban a GNN a személyek és köztük lévő kapcsolatokat elemzi. A csomópontok a személyeket, az élek pedig az olyan kapcsolatokat jelképezik mint a barátság, követés stb. A GNN-ek ezekre a jellemzőkre építve tanulják meg a csomópontokban és az élekben rejlő tulajdonságokat, amiknek ismeretében már képesek az ego hálózatban lévő szociális kapcsolatokban rejlő folyamatok megértésére. Az ilyen ego hálózatokban egy gyakori problémakör a hálózatból hiányzó vagy potenciális kapcsolatok megtalálása, amire egy megoldást jelent az élpredikció. [5]

Fontos kérdés, hogy ezekre a potenciális kapcsolatokra, hogyan lehet rátalálni gráfokban. Az ego hálózatok szerkezete sokszor igen komplex lehet, ezért a gráf méretétől függően az ilyen kapcsolatokat reprezentáló élek megtalálása igen nehéz feladattá válhat. Erre egy megoldást a gráf autoenkóderek (GAE) jelentették, amik a csomópontok jellemzői és a gráf szerkezete között keresnek kapcsolatot. Ezt úgy teszik, hogy a gráf csomóponti jellemzőit egy látens formába alakítják át, amiből egy dekóder segítségével megpróbálják rekonstruálni a gráfot már a potenciális élekkel együtt. Ebben a modellben az autoenkóder funkcióját egy gráf konvolúciós háló tölti be, a dekóder pedig belső szorzattal építi fel újra a gráfot. [6] A további részletek a tanítás fejezetben kerülnek bemutatásra.



1. ábra: GAE modell[1]

3. Használt könyvtárak

Ebben a fejezetben bemutatunk néhány fontosabb könyvtárat, amire a fejlesztés során támaszkodtunk.

- PyTorch: a Facebook Artificial Intelligence Research Group (FAIR) által fejlesztett, a Python nyelvre épülő, magasszintű keretrendszer mélytanulási modellek fejlesztéséhez, betanításához és optimalizálásához. A könyvtár támogatja az NVIDIA GPU használatát, ami lehetővé teszi a nagy adathalmazok kezelését és a mély neurális háló modellek hatékony és gyors betanítását.
- Pytorch Geometrics: a Pytorch keretrendszer kiterjesztése, ami kifejezetten a gráf adatstruktúrák kezelésére és azokon végzett mélytanulásra van specializálva. Ez a könyvtár lehetővé teszi a fejlesztők számára, hogy hatékonyan dolgozzanak gráfkonvolúciós hálózatokkal (GCN) és más gráfon alapuló mélytanulási modellekkel.
- Node2Vec: egy gráf beágyazási módszer, ami lehetővé teszi a pontok (csúcsok) reprezentációját alacsony dimenziós vektorok formájában, amelyek megőrzik a hálózat topológiai információit. A Node2Vec úgy működik, hogy a gráf pontjai közötti sétákat generál, amiket felhasznál a pontok beágyazásához, figyelembe véve a lokális és globális hálózati struktúrákat. Ez a módszer nagy segítséget jelent él predikciós feladatok ellátásában.
- Optuna: egy hiperparaméter optimalizáló Python könyvtár, aminek segítségével gépi tanulási modellekhez könnyedén megtalálhatóak azok a hiperparaméter értékek, amikkel a legnagyobb pontosság elérhető. Használata kifejezetten leegyszerűsíti a hiperparaméter optimalizálását, ugyanis segítségével a teljes folyamat automatizálható.

[1] https://towardsdatascience.com/graph-neural-networks-with-pyg-on-node-classification-link-prediction-and-anomaly-detection-14aa38fe1275

4. Adathalmaz bemutatása

Az adathalmaz a Facebook közösségi háló által összegyűjtött barátok listáját tartalmazza anonimizálva. Az adathalmaz 4039 felhasználó (ego) 88234 kapcsolatát foglalja magába egy gráfként reprezentálva, ami a számos más fájl mellett a nodeId.edges és nodeId.feat fájlból épül fel. A nodeId.edges tartalmazza egy adott ego (nodeId) ismerőseinek kapcsolatait reprezentáló éleket, amik irányítatlanok. A nodeId.feat pedig az ego ismerőseinek jellemzőit írja le. [7]

Az előfeldolgozás során az éleket olyan formátumra kellett hozni, ami lehetővé teszi a gráf szerkezetének vizsgálatát. Az élek ilyen reprezentációjának a segítségével már létrehozható egy gráf, amiből már elkészíthetőek a csomópontok beágyazásai például a Node2Vec beágyazási módszer használatával. Ezeknek a lépéseknek köszönhetően az adathalmaz olyan reprezentációra jutott, amivel modell már tanítható élpredikciós feladatok ellátására.

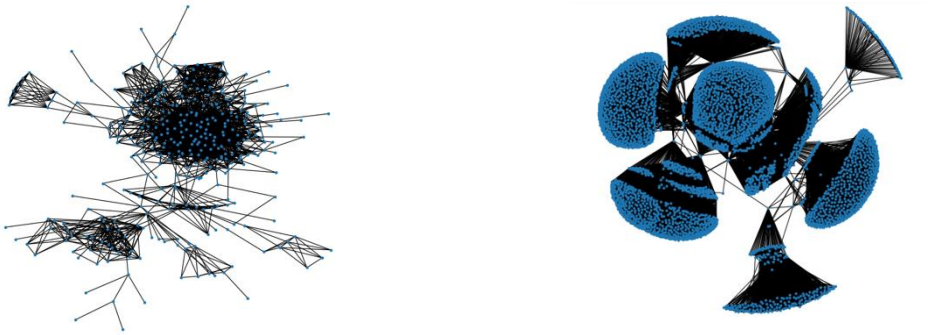
A csomópontok jellemzőin is végeztünk előfeldolgozást azzal a céllal, hogy egy a modell teljesítményét ezekkel a metaadatokkal is javítsuk.

Az előfeldolgozást követően a hiányzó (negatív) élekkel kiegészített adathalmazt felosztottuk a modell tanítási, validációs és tesztelési fázisához. Itt fontos megemlíteni az élek maszkolását. Ez a művelet azért nélkülözhetetlen lépés, mert a tanítás során, amikor a modell a hiányzó kapcsolatokat prediktálja, olyan kapcsolatokat is figyelembe kell vennie, amik az eredeti adathalmazban nem szerepelnek. Ez lehetővé teszi, hogy a tanítás során a modell megtanulja a meglévő éleket megkülönböztetni azoktól, amik potenciálisan jelen lehetnek a gráfban. Ennek köszönhetően a modell képes lesz hatékonyan felismerni és jelezni azokat az éleket, amik valószínűleg léteznek a gráfban. [8]

Az alábbi kódrészletben látható, hogy hogyan oszlanak el az élek és a csúcsok az adathalmaz felosztása után. Az *x* tensor tartalmaza a csúcspontokat és azok jellemzőit, az *edge_index* az összes élet, az *edge_label* azt az értéket (0 vagy 1) hogy a tesztelendő élek valódiak vagy sem, az *edge_label_index* pedig a tesztelendő élek listája, ennek a fele negatív, tehát az eredeti gráfban nem szereplő él.

```
train_data: Data(x=[227, 64], edge_index=[2, 5428], edge_label=[2714], edge_label_index=[2, 2714])
val_data: Data(x=[227, 64], edge_index=[2, 5428], edge_label=[318], edge_label_index=[2, 318])
test_data: Data(x=[227, 64], edge_index=[2, 5746], edge_label=[638], edge_label_index=[2, 638])
```

Az adathalmaz 10 darab kisebb részgráfot tartalmaz, melyek különböző méretűek és felépítésűek. A bal oldali képen látható a teljes Facebook gráf, a jobb oldali képen pedig az első részgráf.



2. ábra: a teljes szociális hálózat toplológiája

5. Tanítás

Ebben a fejezetben a tanítás lépéseit összegeztük. A tanítás fájljai és futtatásának leírása a github repository-inkban található: *https://github.com/TBeatrix/Deep_learning_nagyhazi*

5.1. Tanítás - különböző adatokkal

Az élpredikciót elsőként külön készítettük el mindegyik megadott ego-gráfra és a végén átlagoltuk a teljesítményt, valamint megvizsgáltuk, hogy a különböző tulajdonságú gráfokon melyik módszer hogyan teljesít.

Összehasonlítottuk az él predikció képességét három különböző módszert használva:

- Csak az eredeti csúcspontok tulajdonságait használva
- Csak az általunk generált Node2Vec beágyazott értékeket használva. Ezt úgy valósítottuk meg, hogy a gráf csúcsait reprezentáló X mátrix értékei a Node2Vec eredményeire állítottuk.
- Egyszerre használva a csúcspontok tulajdonságait és a Node2vec beágyazott értékeket. Ezt úgy valósítottuk meg, hogy a gráf csúcsait reprezentáló X mátrix értékeiben minden egyes csúcsnál konkatenáltuk az eredeti jellemzőket és a Node2Vec beágyazásokat.

Megvizsgáltuk és összehasonlítottuk, hogy a különböző konfugációk milyen eredményeket érnek el, ez az Eredmények fejezetben lesz olvasható.

Ezen kívül elkészítettünk egy olyan modellt, amely nem külön-külön tanul a részráfokon, hanem a teljes gráfon fut. Mivel minden egyes részgráfban a csomópontonként megadott tulajdonságok száma eltért, valamint ezek sorrendje és típusa sem volt megegyező, ezek alapján a jellemzők alapján nem tudtunk a teljes gráfon dolgozni.
A node2vec beágyazásoknak, azonban csak az élekre van szüksége, amelyek rendelkezésünkre álltal, így tehát ezt az opciót sikeresen meg tudtuk valósítani a teljes Facebook gráfon.

5.2. Tanítás menete

Általánosságban a használt tanítási függvény megegyezett a fent említett módszereknél, ezt részletezem ebben a fejezetben. Létrehoztunk egy új modellt GNNVAE néven, amely egy gráf autoencodert valósít meg, Az enkóder rész egy kettő rétegű gráf neurális hálózatból áll. Ennek teljesítményét teszteltük gráf konvolúciós hálózat (GCN) esetén és gráf izomorfizmus hálózat

esetén is (GIN). Egy nagyon kis különbséggel a GIN hálózat jobban teljesített, így végeredményben ezt használtuk. Tanítás során az enkóder az eredeti gráfon tanul meg egy reprezentációt (z). Ezután a gráfhoz hozzáadunk véletlenszerű módon annyi random élet, ahány eredeti élünk volt, ezeket nevezzük negatív éleknek. Ehhez létrehozunk egy tömböt amelyben 0 és 1 jelzi, hogy az adott él valós vagy negatív. Az így létrejött gráf tehát fele-fele arányban tartalmaz olyan éleket amelyek valóban szerepelnek a gráfban és olyat, ami nem. Ezt az új gráfot kapja bemenetként a dekóder, melynek feladata eldönteni, hogy az élek közül melyik valós és melyik nem. Gyakorlatileg egy bináris klasszifikációról van szó. Éppen ezért hibafüggvényként is a Bináris Keresztentrópiát használjuk. Tanítás során az eredmény alakulását folyamatosan monitorozzuk a validációs halmazon elért eredménnyel. A validációs halmazban is ugyanannyi negatív él van hozzáadva a gráfhoz, mint amennyi eredeti élt tartalmazott.

5.3. Kiértékelés

A teljes tanítás után, a teszt halmazon is lefuttatjuk a betanított hálózatot és megvizsgáljuk, hogy ezen milyen arányban találta el a valódi éleket. Ennek pontosságát kétféle mérőszámmal adtuk meg: Pontosság és ROC érték.

Emelett megjelenítjük a klasszifikációs riportot, melyben további mérőszámok láthatóak, mint például az f1-score. Megjelenítjük a Konfúziós mátrixot is, hogy látható legyen, hogy az élek vagy a nem élek prediktálásában volt-e jobb a modellünk.

5.4. Hiperparaméterek optimalizálás

A hiperparaméter optimalizálást az Optuna könyvtár segítségével végeztük, ami több paraméter együttest megvizsgált. Az alapján választotta ki a legjobb kombinációt, hogy 80 epoch során hogyan teljesítenek a validációs adathlamazon. A hiperparaméter optimalizálás során a *hidden_channels*, *out_channels* és *learning_rate* értékeire kerestünk egy olyan hármast, amivel a modell lehető legjobb teljesítménye érhető el. Ezt külön kellett elvégezni mindegyik tanítási típusra. Az optimalizálás részletei a final_params_optim.ipynb notebook-ban láthatóak.

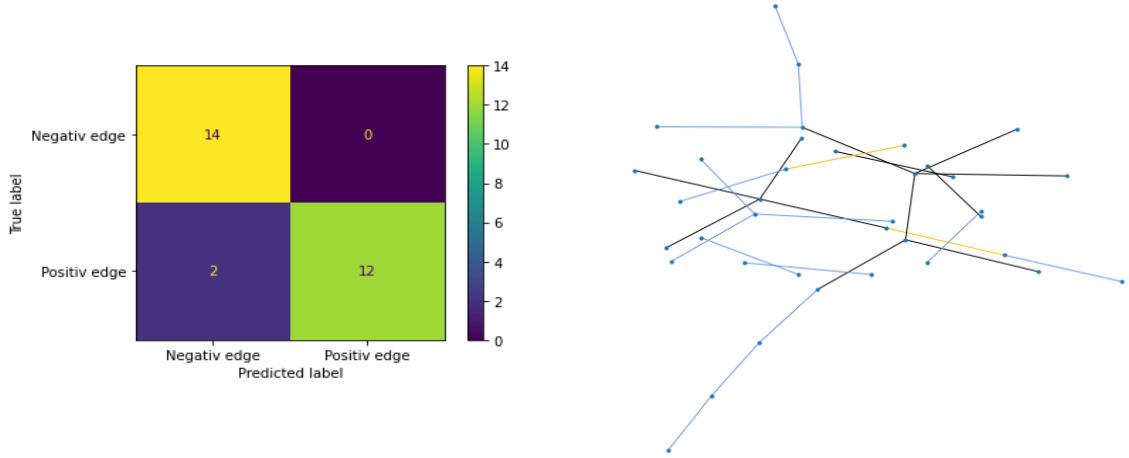
6. Vizualizáció

Mivel a teljes gráf mérete túl nagy lett volna, a vizualizációkban csak a teszthalmaz éleit jelenítjük meg, hogy melyiket prediktálja jól és melyiket nem. A megjelenő élek fele tehát tényleges él, a másik fele pedig negatív él. Ezeket 4 különböző színnel ábrázoltuk:

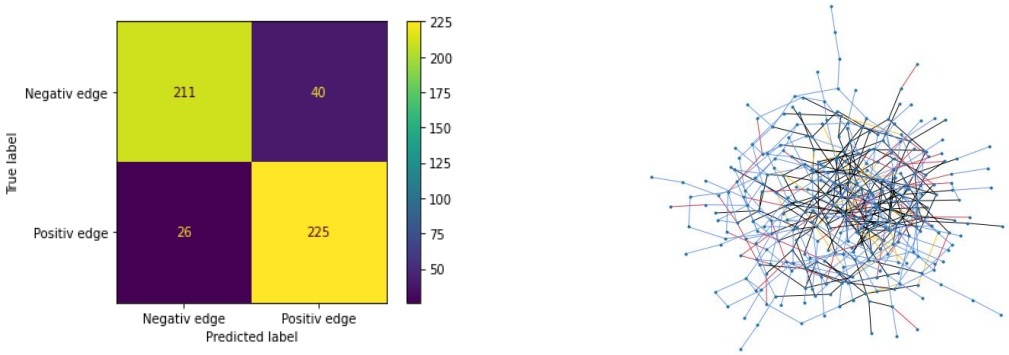
- kék: Azok az élek amelyek az eredeti gráfban is szerepelnek és a rájuk vonatkozó predikció is ezt mondta. **(True Positive)**
- **fekete:** Azok az élek amelyek az eredeti gráfban sem szerepelnek és a rájuk vonatkozó predikció is ezt mondta,hogy ők nem élek. **(True Negative)**
- Piros: Azok az élek amelyek az eredeti gráfban szerepelnek, de a rájuk vonatkozó predikció szerint nem élek. **(False Negative)**

- Narancssárga: Azok az élek amelyek az eredeti gráfban nem szerepelnek, de a rájuk vonatkozó predikció szerint igen. **(False Positive)**

Ezek az ábrák szemléletessé teszik, hogy melyik kapcsolatokat tudta a gráf megtanulni, ezekben mintázatok kereshetőek, hogy esetleg mit nem képes megtalálni a gráf. A számszerű kiértékelések mellett így érdemes ezeket is figyelembe venni. A nagyobb gráfokon, sajnos ez a módszer nem használható, mert ahogy az a notebookban is megfigyelhető, ezeken az élek nagy száma miatt, nem lehet őket megkülönböztetni egymástól. A kis gráfokon azonban szépen látszik az eredmény. Itt látható egy-egy példa a 10. és az 1. részgráfról:



3. ábra: 10. Részgráf konfóziós mátrixa és topológiája



4. ábra: Az 1. Részgráf topplógiája

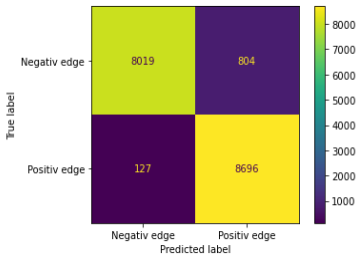
7. Eredmények

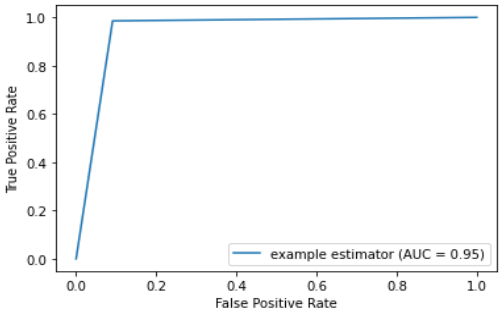
A különböző variációk legjobb eredményei:

	AUC score	Accuracy
Teljes gráf (Node2Vec)	**0.981**	**0,947**
Részgráfok átlaga (Node2Vec)	**0.905**	**0.854**
Részgráfok átlaga (Node features)	**0.882**	**0.845**
Részgráfok átlaga (Node features + Node2Vec)	**0.906**	**0.865**

A **dl_final.ipynb** notebook tartalmazza az összes gráf összes tanításához tartozó Klasszifikációs reportot, konfúziós mátrixot és a gráf színezett teszt adathalmazbeli éleiről készített vizualizációt. Ez utóbbira az előző fejezetben látható példa. A teljes gráfhoz tartozó kiértékelés a következő képeken látható:

	precision	recall	f1-score	support
0	0.98	0.91	0.95	8823
1	0.92	0.99	0.95	8823
accuracy			0.95	17646
macro avg	0.95	0.95	0.95	17646
weighted avg	0.95	0.95	0.95	17646





Eredmények összesítése:

A részgráfokon való teljesítményt ábrázoltuk mindhárom módszerrel, Pontosság és AUC érték szempontjából. Ezek láthatóak a következő diagrammokon. (Az y tengelyen csak a 0.7 feletti részt jelenítjük meg, hogy az eltérések megfelelően láthatóak legyenek)

Diagram az AUC értékekről (kék: Node features + Node2Vec, narancssárga: Node2Vec, zöld:Node features):

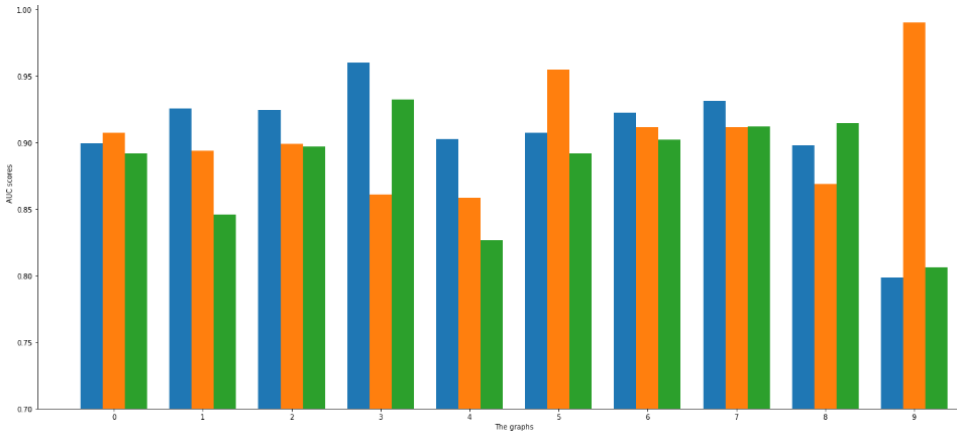
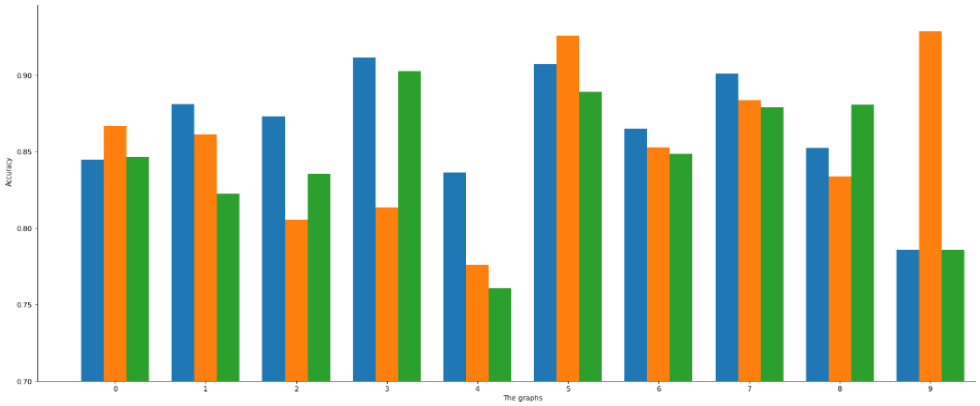


Diagram a Pontosság értékekről (kék: Node features + Node2Vec, narancssárga: Node2Vec, zöld:Node features):



Konlúziók:

- A **legjobban a teljes gráfon teljesít az algoritmus**, hiszen itt rengeteg éle van a gráfnak, melyekből sikeres meg tudja tanulni az algoritmus a megfelelő műveletet.
- A részgráfok közül nehéz meghatározni, hogy melyik módszer volt a legjobb. Ha az átlagos teljesítményt nézzük, akkor az a variáció teljesített a legjobban amikor a csúcsok beágyazásában az eredeti csúcs jellemzőket is használtuk, valamint a Node2Vec aloritmussal előállított strukturális információkat is. Mivel ez mindkét másik módszer egyesítése, ettől vártuk mi is a legjobb megoldást.
- Azonban megfigyelhető a diagramokon, hogy nem minden részgráfon ez a legjobban teljesítő algoritmus (kék oszlop).
- Az az összefüggés figyelhető meg hogy ha egy gráf kicsi, akkor a csúcs jellemzőit használó algoritmus éri el a legjobb teljesítményt (Narancssárga) Ez figyelhető meg a két legkisebb gráfon: a 6.on és a 10.-en.
- Viszont, ha a gráf nagy, akkor a Node2Vec-et használó algoritmus teljesít jobban. (Zöld oszlop). Ez figyelhető meg például a 4. és a 9. gráfon. Ebből levonható az a következtetés, hogy kisebb gráfokon érdemesebb a csúcsok jellemzőit használni, mivel itt a gráf kis méretéből adódóan a gráf szerkezetéből számított információk még nem jelentősek. Azonban minél nagyobb a gráf, annál jelentősebbé válnak és végül túl is szárnyalják a Node2vec beágyazások a csúcsjellemzőket.
- Érdemes tehát mindegyik módszert tesztelni, de általánosan az a legjobb megoldás, amikor mind a kettőt (csúcs jellemzők + Node2vec beágyazások) egyszerre használjuk.
- A konfúziós mátrixokon az látható, hogy nagyobb számban fordulnak elő a True Positive-ok mint a False Negative-ok. Tehát inkább mondunk egy nem létező élet létezőnek, mint fordítva. Ez utalhat arra, hogy a gráfban lehet, hogy vannak olyan ismeretségek amik fent állnak, de még sincsenek jelölve, ezek potenciális kapcsolatok lehetnek. Új lehetséges kapcsolatokat tehát úgy találhatnánk, hogy megnézük a legmagasabb valószínűséggel False Positive éleket.

Referenciák

[1] Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022, December 3). Graph Neural Networks in Recommender Systems: A Survey. *ACM Computing Surveys*, *55*(5), 1–37. https://doi.org/10.1145/3535101

[2] Jiang, D., Wu, Z., Hsieh, C. Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., & Hou, T. (2021, February 17). Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, *13*(1). https://doi.org/10.1186/s13321-020-00479-8

[3] Scarselli, F., Gori, M., Ah Chung Tsoi, Hagenbuchner, M., & Monfardini, G. (2009, January). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. https://doi.org/10.1109/tnn.2008.2005605

[4] Ahn, Y. Y., Bagrow, J. P., & Lehmann, S. (2010, June 20). Link communities reveal multiscale complexity in networks. *Nature*, *466*(7307), 761–764. https://doi.org/10.1038/nature09182

[5] Mcauley, J., & Leskovec, J. (2014, February). Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery From Data*, *8*(1), 1–28. https://doi.org/10.1145/2556612

[6] Kipf, T. N., & Welling, M. (2016, November 21). Variational graph auto-encoders. Bayesian Deep Learning Workshop (NIPS 2016). arXiv:1611.07308 [stat.ML]. https://doi.org/10.48550/arXiv.1611.07308

[7] Mcauley, J., & Leskovec, J. (2014, February). Discovering social circles in ego networks. ACM Transactions on Knowledge Discovery From Data, 8(1), 1–28. https://doi.org/10.1145/2556612

[8] Masui, T. (2022, October 17). Graph Neural Networks with PyG on Node Classification, Link Prediction, and Anomaly Detection. Medium. https://towardsdatascience.com/graph-neural-networks-with-pyg-on-node-classification-link-prediction-and-anomaly-detection-14aa38fe1275