Analiza vplivov kodirnikov na doseženo okrepitveno nagrado hibridnega algoritma NARM-XCS

Damijan Novak, Domen Verber, Iztok Fister, Iztok Fister ml.

Inštitut za informatiko, Univerza v Mariboru, Koroška Cesta 46, 2000 Maribor, Slovenija
damijan.novak@um.si, domen.verber@um.si, iztok.fister@um.si, iztok.fister1@um.si

Influence of Encoder Selection on Acquired Reward in a Hybrid NARM-XCS Algorithm

Abstract. *The ever-increasing demand for processing large datasets requires more efficient data utilization. Reusing valuable information previously processed by a Numerical Association Rule Mining (NARM) algorithm promotes energy efficiency. Furthermore, combining NARM with the reinforcement learning eXtended Classifier System (XCS) algorithm enhances the system's adaptability. This work analyzes a hybrid NARM-XCS algorithm, focusing on how the encoders influence the algorithm's reward gain. The experiments compared the performance of three encoders (Binary Encoder, KMeans Encoder, and One-Hot Encoder) on health, medicine, and game domain datasets. Understanding this influence can help optimize the algorithm's performance even further.*

1 Uvod

S prehodom v poznih sedemdesetih letih iz industrijske dobe v informacijsko dobo se je družba preoblikovala ekonomsko, socialno ter kulturno [1]. Vzporedno so potekali (in še potekajo) tudi prehodi od Industrije 3.0 temelječe na avtomatizaciji in elektroniki v Industrijo 4.0, ki se primarno navezuje na raziskovalna področja pametnih tovarn ter interneta stvari. Vključuje tudi pod področja kot so kiberfizični sistemi (npr. robotika ter znanost *umetne inteligence* (ang. *Artificial Intelligence*, AI)) [2]. Na prehod v Industrijo 5.0 se že pripravljamo. V njej bo poudarek na sodelovanju med ljudmi in stroji, s ciljem izboljšanja človekovega počutja [3].

Opazimo lahko, da so inteligentne tehnike, ki so širšim množicam bolj poznane pod pojmom AI, ključnega pomena za družbo. Znotraj tehnik AI ima pomembno vlogo strojno učenje, zlasti učenje na podlagi nagrade, katerega mehanizme v tem članku učinkovito izkoriščamo. Strokovno se to področje strojnega učenja imenuje *okrepitveno učenje* (angl. *reinforcement learning*, RL).

RL predstavlja tip učenja znan pod izrazom *poskus-napaka* (angl. *trial-and-error*), saj mora agent na osnovi izvedbe akcij nad okoljem odkriti katere akcije mu prinašajo največjo nagrado [4]. Povedano še drugače, uspešen poskus prinaša pozitivno nagrado, neuspešen oz. napaka pa negativno, pri čemer je možna tudi nevtralna vrednost/nagrada nič.

V članku uporabimo dva napredna algoritma strojnega učenja poimenovana *numerično rudarjenje asociativnih pravil* (angl. *association rule mining*, NARM) ter *razširjen sistem na osnovi klasifikatorjev* (angl. *extended classifier system*, XCS). Oba algoritma sta bila prehodno združena v hibrid v izvirnem znanstvenem prispevku, ki je podrobno opisan in predstavljen pod referenco [5], ter ki vključuje tudi sliko celotne arhitekture hibrida NARM-XCS. Glavni poudarek prispevka je bil na simbiozi obeh algoritmov z namenom ponovne uporabe pravil NARM-a, na povezavi informacij pravil NARM z akcijami sistema XCS, ter na prilagajanju obstoječih NARM pravil novim stanjem okolja. Hibrid je za namen raziskave v osnovni verziji uporabljal osnovni binarni kodirnik (angl. binary encoder). V tem delu pa bomo uporabili še napredna kodirnika poimenovana kodirnik k-Means (tj., algoritem za kodiranje kategorij na podlagi k-means gručenja [6]) ter kodirnik One-Hot (tj., algoritem za kodiranje kategoričnih spremenljivk v binarne vektorje) za namen analize vpliva izbire kodirnika na okrepitveno nagrado, ki jo pridobi hibrid med svojim delovanjem.

Prispevek je strukturiran tako, da v drugem poglavju predstavimo teoretično ozadje vseh algoritmov in metod potrebnih za izvedbo raziskave. V tretjem poglavju se nahaja opis in rezultati eksperimenta. V zadnjem zaključnem poglavju pa je podana kratka diskusija in povzetek dela.

2 Teoretično ozadje

V poglavju 2 se bomo osredotočili na vse metode, ki so bile uporabljene v kontekstu raziskave. Opisana sta algoritma NARM ter XCS, in vsi uporabljeni kodirniki.

2.1 Numerično rudarjenje asociativnih pravil

Rudarjenje asociativnih pravil (angl. *Association Rule Mining*, ARM) je tehnika strojnega učenja, ki se uporablja za odkrivanje skritih vzorcev, povezav ali zakonitosti v velikih podatkovnih zbirkah. Ključni koncepti za ARM vključujejo metrični oceni kot sta *podpora* (angl. *support*) in *zaupanje* (angl. *confidence*) (tj., obstajajo tudi druge metrike), ter omogočata oceno pomembnosti in uporabnosti odkritih pravil [7].

ARM je bil sprva usmerjen v uporabo zgolj nad kategoričnimi podatki, a se je nato razširil še nad numerično področje (tj., NARM) [8]. NARM je tako zasnovan za iskanje povezav med numeričnimi atributi v podatkovnih zbirkah relacij. Podatkovna zbirka je sestavljena iz relacij, vsaka relacija pa vsebuje podmnožico vseh možnih vrednosti atributov. Vsak atribut je lahko določen z intervalom realnih vrednosti ali pa s kategorijami.

2.2 Razširjen sistem na osnovi klasifikatorjev

Učeči se sistemi na osnovi klasifikatorjev (*Learning Classifier Systems*, LCS) so družina algoritmov strojnega učenja, ki se uporabljajo za reševanje problemov klasifikacije, napovedovanja in optimizacije [9]. Temeljijo na uporabi populacije klasifikatorjev. Klasifikatorji se sčasoma izboljšujejo na podlagi izkušenj. S prvo predstavitvijo LCS svetovni raziskovalni skupnosti, je ta postala bogatejša tudi za (sedaj zelo znan) genetski algoritem, ki je namreč osrednja komponenta LCS algoritma [10]. S kombinacijo genetskega algoritma ter RL se doseže izboljševanje populacije klasifikatorjev, čemur tudi pravimo učenje klasifikatorjev.

V tem delu uporabljamo XCS algoritem, saj je ta znan po visoki *prilagodljivosti* [11] (angl. *adaptivity*) klasifikatorjev *v času delovanja* (angl. *on-line learning*) [12], ker proizvede maksimalno *generalizirane* (angl. *generalisation*) klasifikatorje [13], ter ker se ga da kvalitetno uporabiti za množico različnih raziskovalnih problemov [14].

Klasifikator v sistemu XCS je preprosto pravilo v obliki ‘IF stavka’, sestavljeno iz pogoja (v binarni obliki) in akcije. Če se pogoj izpolni, se akcija izvede tekom glavnega cikla (tj., vmes ko teče iterativna glavna zanka) sistema XCS. Pogoj lahko vsebuje tudi simbol ‘#’ ("ne zanima me"), ki omogoča klasifikatorju, da se posploši na vhode in se tako uči širšega spektra stanj okolja.

2.3 NARM-XCS

Tekom našega predhodnega dela je bila ustvarjena osnovna struktura hibrida algoritma NARM-XCS. Algoritem deluje tako, da se najprej vnaprej procesirana pravila NARM-a pretvorijo v klasifikatorje, ki jih lahko algoritem XCS uporabi. To se izvede s pomočjo kodirnika, ki informacije kodira v binarne vrednosti. Tako ustvarjeni kodirani klasifikatorji se nato vključijo v algoritem XCS, ki jih preko evolucijskih mehanizmov nato nadaljnje izpopolnjuje s pomočjo okolja (tj., nad učno množico podatkovne zbirke), komponent RL in cenitvene funkcije, da se najde optimalna akcija (tj., izbrani atribut, ki je lahko ali atribut, ki predstavlja razred vrstice, ali pa je to poljubno izbrani atribut). Preverjanje kvalitete naučene populacije klasifikatorjev pa se nato izvede s testno množico podatkovne zbirke. Če je pridobljena nagrada ustrezna (tj., se približamo maksimalni meji nagrade cenitvene funkcije), lahko rečemo, da je bilo učenje uspešno.

Kodirniki pa za namen delovanja cenitvene funkcije omogočajo tudi kodiranje podatkov podatkovne zbirke (tj., posamezne vrstice zbirke). Na ta način se lahko v cenitveni funkciji izvaja ustrezna primerjava vhodnih podatkov okolja s klasifikatorji algoritma XCS.

2.4 Kodirniki

Za namene raziskave smo se omejili na tri metode s katerimi kodiramo podatke. In sicer na, binarni kodirnik, k-means kodirnik ter One-Hot kodirnik Opomba: One-Hot kodirnik je še dodatno nadgrajen z elementi binarnega kodirnika.

Razlog za izbiro teh kodirnikov leži v tem, da so izredno razširjeni ter tudi učinkoviti pri pretvorbi podatkov v oblike primerne za računalniško obdelavo (npr. kot podpora za delo s podatki pri algoritmu ARM-XCS). Na primer, osnovna verzija binarnega kodirnika se je že pokazala kot zelo učinkovita v naših predhodnih raziskavah [5]. k-means kodirnik je vključen zaradi njegove zmožnosti razdeljevanja podatkov v skupine in njihovega predstavljanja v strukturirani obliki. Prilagojen One-Hot kodirnik pa nam omogoča kombinacijo prednosti enostavnega binarnega kodiranja in večje natančnosti zaradi dodatne binarne pretvorbe.

Binarni kodirnik je zmogljiva metoda za pretvorbo numeričnih vrednosti v binarno obliko. Osrednji koncept te metode je razdelitev atributov podatkovne zbirke v naprej določeno število intervalov oz. razdelkov (angl. bin). Metoda obdeluje numerične podatke tako, da razvrsti vrednosti atributov v razdelke, pretvori te razdelke v nize binarnih števk z določeno dolžino ter jih združi v kodirani seznam. Naša implementacija binarnega kodirnika pa poleg vrstic podatkovne zbirke podpira tudi kodiranje pravil NARM. Ker pravila NARM podobno kot vrstice podatkovne zbirke tudi temeljijo na atributih ter vrednostih, je takšno kodiranje mogoče.

Kodirnik k-means je algoritem, ki kodira numerične podatke z uporabo metode k-means (tj., k-srednjih) vrednosti. Za vsak atribut izračuna pripadajočo kategorijo z uporabo modela k-means vrednosti, pretvori to kategorijo v niz binarnih števk ustrezne dolžine, in jih združi v kodirani seznam.

One-Hot kodirnik se uporablja za kodiranje kategoriziranih podatkov v binarne oblike. Pred začetkom kodiranja se inicializira podatkovna struktura slovarja kategorij, kjer vsak stolpec v slovarju dodeli seznam edinstvenih vrednosti. Med kodiranjem se nato vsako vrednost v stolpcu pretvori v niz bitov, pri čemer je število bitov določeno glede na število kategorij v stolpcu. Izračun števila bitov poteka po formuli (1)

$$\text{številoBitov} = \lceil \log_2(\text{velikost}(\text{slovarKategorij})) \rceil \quad (1)$$

kjer velikost(slovarKategorij) predstavlja število različnih kategorij, formula pa zagotavlja, da je vsaka kategorija pravilno kodirana v ustrezno število bitov. Za zmanjšanje dimenzionalnosti končne kodirane predstavitve pri kodirniku One-Hot uporabimo tehniko zgoščevanja (tj., z uporabo zgoščene tabele). Namesto da bi neposredno uporabljali število kategorij za določitev števila bitov, se nad vsako vrednostjo kategorije uporabi funkcija zgoščevanja. Rezultat zgoščevanja se nato uporabi za nastavitev podmnožice bitov v končnem nizu kodiranih bitov, kar učinkovito zmanjša potrebno število bitov za kodiranje, ne da bi pri tem izgubili pomembne informacije o kategoriji.

3 Eksperiment in rezultati

Za namen eksperimenta so bile izbrane tri podatkovne zbirke iz repozitorija kaggle (vse zbirke so izdane v letu 2024): "Air quality health impact" (AQHI) (slov. vpliv zraka na kakovost zdravja) [15] z 5.811 vrsticami, "Breast cancer dataset" (BCD) [16] (zbirka podatkov za bolezen raka dojk) z 569 vrsticami, ter "League of Legends SoloQ matches at 15 minutes 2024" (LOL) (zbirka podatkov, ki se nanaša na statistične podatke iz igre League Of Legends) [17] z 24.225 vrsticami. Iz vsake zbirke pa je najprej bil odstranjen prvi stolpec, ki predstavlja ID vrednost, saj ti podatki niso nosilci relevantnih informacij za naš eksperiment.

Pri vsaki podatkovni zbirki imamo možnost izbrati poljuben atribut nad katerim operira naš hibridni algoritem (tj., algoritem ni omejen samo na en atribut, ki bi na primer klasificiral podatkovno vrstico zbirke v specifičen razred kot je to običajno pri klasifikacijskih algoritmih). Za vsako podatkovno zbirko smo tako naključno izbrali en atribut. In sicer za zbirko AQHI je bil izbran atribut *PM10*, za BCD atribut *fractalDimensionWorst*, ter za LOL atribut *blueTeamTurretPlatesDestroyed*.

Nastavitve vrednosti hiperparametrov algoritma NARM ter XCS so enake kot v članku prve predstavitve hibrida NARM-XCS [5]. Na tak način se zagotavlja večja sledljivost in primerjava obeh eksperimentov.

Eksperiment je bil izveden na računalniku z vsebovanim s procesorjem i7-9700 (osem jeder), 32 GB sistemskega pomnilnika, na operacijskem sistemu OS Windows 10 Pro z razvojnim orodjem IntelliJ IDEA 2024.1.4 (Ultimate Edition) ter v programskem jeziku Java (verzija 22).

Za namen ustvarjanja pravil je bilo procesiranje NARM izvedeno samo enkrat za vsako podatkovno zbirko. Pri ponovitvah eksperimentalnih testov tako izviramo iz iste osnove pravil NARM, kar nam zagotavlja, da je eksperiment dinamično odvisen samo od kodirnikov ter od procesiranja z algoritmom XCS (tj., procesiranje NARM običajno proizvede vsakič malenkost drugačna pravila zaradi prisotne naključnosti v algoritmu diferencialne evolucije; v tem primeru pa je sedaj zadeva statična).

Posamezne nagrade se vrnejo s pomočjo *cenitvene funkcije* (angl. *evaluation function*). Okrepitvena nagrada je zasnovana tako, da funkcija vrne nagrado v vrednosti dve, če je algoritem XCS izbral akcijo (tj., atribut), ki smo jo izbrali kot privzeto (tj., atribut, ki je izbran za prilagajanje populacije klasifikatorjev z algoritmom XCS). V nasprotnem primeru se vrne negativna nagrada v vrednosti minus ena. Skupna nagrada se izračuna tako, da se vsota vseh prejetih nagrad deli s številom vrstic podatkovne zbirke (tj., učne ali pa testne zbirke).

Eksperiment je potekal tako, da se je nad vsako podatkovno zbirko pognalo vse tri kodirnike po petkrat, in sicer po sistemu, da je imel binarni kodirnik enkrat število razdelkov nastavljeno na dva (b2) in enkrat na tri (b3), ter kodirnik k-means enkrat z opcijo k je dva (k2) in drugič z opcijo k je tri (k3) (Opomba: Če za določitev k skupin ni na voljo zadostno število edinstvenih podatkovnih točk je v tabeli zapisana oznaka N/A). Število ponovitev algoritma k-means je nastavljeno na maksimalno 100. Kodirnik One-Hot pa ne potrebuje posebne začetne nastavitve vrednosti hiperparametrov.

V tabelah 1, 2 in 3 so podani povprečni rezultati doseženih okrepitvenih nagrad, in sicer enkrat z izključeno opcijo vstavljanja pravil NARM v populacijo XCS algoritma, ter drugič z vključeno opcijo vstavljanja pravil (tj., do sto procentov pravil populacije se lahko zamenja s pravili NARM).

Tabela 1. Podatki za podatkovno zbirko AQHI.

Kodirnik (procent vstavljenih pravil, zbirka podatkov [učna zbirka: uz, testna zbirka: tz])	**Nagrada**
Binarni kodirnik_b2 (0, uz)	1,778
Binarni kodirnik_b3 (0, uz)	1,738
Kodirnik k-means_b2 (0, uz)	1,812
Kodirnik k-means_b3 (0, uz)	1,826
Kodirnik One-Hot (0, uz)	1,784
Binarni kodirnik_b2 (0, tz)	1,817
Binarni kodirnik_b3 (0, tz)	1,75
Kodirnik k-means_k2 (0, tz)	1,865
Kodirnik k-means_k3 (0, tz)	1,865
Kodirnik One-Hot (0, tz)	1,873
Binarni kodirnik_b2 (100, uz)	1,869
Binarni kodirnik_b3 (100, uz)	1,863
Kodirnik k-means_k2 (100, uz)	1,854
Kodirnik k-means_k3 (100, uz)	0,372
Kodirnik One-Hot (100, uz)	1,798
Binarni kodirnik_b2 (100, tz)	1,893
Binarni kodirnik_b3 (100, tz)	1,896
Kodirnik k-means_k2 (100, tz)	1,872
Kodirnik k-means_k3 (100, tz)	1,672
Kodirnik One-Hot (100, tz)	1,877

Tabela 2. Podatki za podatkovno zbirko BCD.

Kodirnik (procent vstavljenih pravil, zbirka podatkov [učna zbirka: uz, testna zbirka: tz])	**Nagrada**
Binarni kodirnik_b2 (0, uz)	1,676
Binarni kodirnik_b3 (0, uz)	1,875
Kodirnik k-means_b2 (0, uz)	1,707
Kodirnik k-means_b3 (0, uz)	N/A
Kodirnik One-Hot (0, uz)	1,84
Binarni kodirnik_b2 (0, tz)	1,532
Binarni kodirnik_b3 (0, tz)	1,868
Kodirnik k-means_k2 (0, tz)	1,611
Kodirnik K-means_k3 (0, tz)	N/A
Kodirnik One-Hot (0, tz)	1,795
Binarni kodirnik_b2 (100, uz)	1,739
Binarni kodirnik_b3 (100, uz)	1,854
Kodirnik k-means_k2 (100, uz)	1,761
Kodirnik k-means_k3 (100, uz)	N/A
Kodirnik One-Hot (100, uz)	1,821
Binarni kodirnik_b2 (100, tz)	1,726
Binarni kodirnik_b3 (100, tz)	1,847
Kodirnik k-means_k2 (100, tz)	1,663
Kodirnik k-means_k3 (100, tz)	N/A
Kodirnik One-Hot (100, tz)	1,816

Tabela 3. Podatki za podatkovno zbirko LOL.

Kodirnik (procent vstavljenih pravil, zbirka podatkov [učna zbirka: uz, testna zbirka: tz])	Nagrada
Binarni kodirnik_b2 (0, uz)	1,835
Binarni kodirnik_b3 (0, uz)	-0,953
Kodirnik k-means_b2 (0, uz)	1,661
Kodirnik k-means_b3 (0, uz)	N/A
Kodirnik One-Hot (0, uz)	-0,322
Binarni kodirnik_b2 (0, tz)	1,869
Binarni kodirnik_b3 (0, tz)	-0,992
Kodirnik k-means_k2 (0, tz)	1,843
Kodirnik k-means_k3 (0, tz)	N/A
Kodirnik One-Hot (0, tz)	1,314
Binarni kodirnik_b2 (100, uz)	1,85
Binarni kodirnik_b3 (100, uz)	-0,939
Kodirnik k-means_k2 (100, uz)	1,86
Kodirnik k-means_k3 (100, uz)	N/A
Kodirnik One-Hot (100, uz)	0,588
Binarni kodirnik_b2 (100, tz)	1,874
Binarni kodirnik_b3 (100, tz)	-0,991
Kodirnik k-means_k2 (100, tz)	1,867
Kodirnik k-means_k3 (100, tz)	N/A
Kodirnik One-Hot (100, tz)	1,87

4 Diskusija in zaključek

S tem člankom smo izvedli analizo vpliva različnih kodirnikov na doseženo nagrado hibridnega algoritma NARM-XCS. Rezultati so pokazali izredno odpornost hibridnega algoritma na tip izbranega kodirnika, saj so vsi trije dosegali vrednosti nagrad blizu zgornjim mejam, ki jih lahko zagotovi osnovni algoritem XCS (t.j., z izjemami za kodirnik k-means_k3 za AQHI, in binarni kodirnik_b3 ter One-Hot za zbirko LOL).

Med primerjavo rezultatov tega članka z rezultati v originalnem članku predstavljenega algoritma NARM-XCS, smo opazili, da je razlika v nagradah brez dodanih pravil in pri sto procentov vstavljenih pravil NARM v XCS pri teh podatkovnih zbirkah zaznavna (tj., v prid vstavljanja pravil), vendar je manjšega obsega kot pri dveh zbirkah uporabljenih v originalnem članku.

Z informacijami pridobljenimi v tej analizi imamo tako izvrstno popotnico za možnosti raziskav še v prihodnje. In sicer, preučiti kakšna je povezava specifične zbirke podatkov (tj., glede števila vrstic zbirke in razporeditve podatkov) z različnimi vrednosti hiperparametrov (npr. velikost populacije XCS), ter nato še dodatno kakšna je pri tem povezava vplivov različnih procentov vstavljenih pravil NARM v populacijo XCS algoritma (tj., raziskati kdaj se to vstavljanje obrestuje, ter odkriti če je kakšen specifičen vzorec podatkovnih zbirk za katere se hibridni algoritem še posebej izkaže kot izredno dobra izbira).

Zahvala

Raziskovalni program št. P2-0057 je sofinancirala Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije iz državnega proračuna.

Literatura

[1] M. Castells, C. Blackwell: The information age: economy, society and culture, Volume 1. The rise of the network society. Environment and Planning B: Planning and Design, 25, str. 631-636, 1998.

[2] K. H. Tantawi, A. Sokolov, O. Tantawi: Advances in industrial robotics: From industry 3.0 automation to industry 4.0 collaboration, In 2019 4th TIMES-iCON, str. 1-4, IEEE, 2019.

[3] J. Leng, W. Sha, B. Wang, P. Zheng, C. Zhuang, Q. Liu, et al.: Industry 5.0: Prospect and retrospect, Journal of Manufacturing Systems, 65, str. 279-295, 2022.

[4] R. S. Sutton, A. G. Barto: Reinforcement learning: An introduction, MIT press, 2018

[5] D. Novak, D. Verber, I. Fister, I. Fister Jr.: Association Rule Mining as Knowledge Infusion into the Mechanics of an eXtended Classifier System, 2024 IEEE 28th INES, str. 203-208, 2024.

[6] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu: An efficient k-means clustering algorithm: Analysis and implementation, IEEE transactions on pattern analysis and machine intelligence, 24(7), str. 881-892, 2002.

[7] R. Agrawal, R. Srikant: Fast algorithms for mining association rules, In 20th int. conf. very large databases, VLDB, Morgan Kaufmann Publishers Inc., zv. 1215, str. 487-499, 1994.

[8] I. Fister, A. Iglesias, A. Galvez, I. Jr. Fister: Online numerical association rule miner, Neurocomputing, Elsevier, zv. 523, str. 33-43, 2023.

[9] M. V. Butz: Learning classifier systems, Springer handbook of computational intelligence, str. 961-981, 2015.

[10] R. E. Smith: Learning classifier systems, In Evolutionary Computation 1, CRC Press, str. 152-161 2018.

[11] P. L. Lanzi: Learning classifier systems: then and now, Evolutionary Intelligence, zv. 1, str. 63-82, 2008.

[12] N. Fredivianus, H. Prothmann, H. Schmeck: XCS revisited: a novel discovery component for the eXtended classifier system, In Asia-Pacific Conference on Simulated Evolution and Learning, Berlin, Heidelberg: Springer Berlin Heidelberg, str. 289-298, 2010.

[13] A. R. Wagner, A. Stein: Mechanisms to alleviate over-generalization in XCS for continuous-valued input spaces, SN Computer Science, 3(2), 176, 2022.

[14] A. Stein, R. Maier, L. Rosenbauer, J. Hähner: XCS classifier system with experience replay, In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, str. 404-413, 2020.

[15] Rabie El Kharoua, Air Quality and Health Impact Dataset Kaggle, 2024. https://doi.org/10.34740/KAGGLE/DSV/8675842

[16]Breast Cancer Dataset, 2024. https://www.kaggle.com/datasets/krupadharamshi/breast-cancer-dataset

[17] League of Legends SoloQ matches at 15 minutes 2024, 2024. https://www.kaggle.com/datasets/karlorusovan/league-of-legends-soloq-matches-at-10-minutes-2024/data