# Synthetic Minority Oversampling Technique based on Probability Distribution (SyMProD)[[1]](#footnote-1)

Տվյալագիտության մեջ արդի խնդիր է համարվում ոչ բալանսավորված տվյալների առկայությունը դասակարգման խնդիրների լուծման համար։ Խնդիրն այն է, որ մոդելը լավ է գուշակում մեծամասնության խմբին պատկանող տողերը և վատ է գուշակում փոքրամասնության խմբի տողերը և հիմնականում խնդիրը կայանում է նրանում, որ դասակարգման խնդիրների ժամանակ առավել կարևոր է լինում թե մոդելը ինչ ճշգրտությամբ կգուշակի փոքրամասնության խմբին պատկանող տողերը, որպես օրինակ՝ հիվանդությունների հայտնաբերման, սպամ նամակների հայտնաբերման, կրեդիտ քարտով կասկածելի գործարքի հայտնաբերումը զեղծարարությունից խուսափելու համար։ Ոչ բալանսավորված տվյալների խնդիրը հաղթահարելու համար ժամանակակից տվյալագիտությունը առաջարկում է Սինթետիկ ձևով նոր տողեր գեներացնելու տարբեր ալգորիթմներ։ Այդ ալգորիթմների մեծամասնությունը կարող է մեծացնել մոդելի գուշակման ճշգրտությունը, սակայն մեծամասնությունը չի ուսումնասիրում թե ինչպիսի բաշխում ունեն փոքրամասնության դասին պատկանող տողերը։ Synthetic Minority Oversampling Technique based on Probability Distribution (SyMProD) ալգորիթմը ուսումնասիրում է այդ խնդիրը և նախքան նոր տողերը գեներացնելը դուրս է բերում օտար/outlier համարվող տվյալները և նոր գեներացվող տողը ստանում է որոշակի պատահական ընտրված տողի և նրան մոտակա հարևան տողերի կշռավորված գծային կոմբինացիայից։

Քայլ առ քայլ ներկայացնենք թե մեթոդը ինչպես է աշխատում և ամեն քայլին ինչ խնդիր է լուծվում։ Սկզբնապես որոշվում է թե ինչ քանակի նոր տողեր պետք է գեներացվեն, որպեսզի տվյալները դասերի քանակները հավասարվեն։

nmaj – nmin = ngen

1. Աղմուկի ֆիլտրում․

Աղմուկի ֆիլտրեր նշանակում է ձերբազատվել օտար համարվող տվյալներից, սակայն սա չի կատարվում քվանտիլների մեթոդով, այլ տվյալների բոլոր էլեմենտները նորմավորվում են ըստ սյունի, այսինքն՝

Ալգորիթմը ունենում է սկզբնապես տրված պարամետրեր, որոնցից մեկը Աղմուկի Շեմն է Noise Threshold(NT): Տողերը ֆիլտրվում են հետևյալ կերպ՝ եթե , ապա այդ տողը կբացառվի, ընդ որում պետք է հաշվի առնել, որ x’-ը իրենից ներկայացնում է տող և եթե իր սյուներից մեկի արժեքը գերազանցի Աղմուկի Շեմը ապա այդ տողը կբացառվի։

1. Բացառել Overlapping-ի խնդիրը

Այս քայլը կարևոր է այն պատճառով, որ բացի օտար/outlier համարվող տողերից կան նաև այնպիսի տողեր, որոնք լինում են շրջապատված այլ մեծամասնության դասին պատկանող տողերով, ուստի անհրաժեշտ է բացառել դրանք նույնպես, որպեսզի այդ կետերը նոր գեներացվող տողերի մեջ չմասնակցեն։ Քանի որ իրական տվյալների հիմնականում գծորեն անջատելի չեն, և հիմնականում նրանք խմբերով կուտակված չեն լինում մի ֆիքսված տեղ, ուստի տողը բացառելու այս քայլում էլ օգտագործվում է Կտրման Շեմը/ Cutoff Threshold-ը (CT)։ Նախորդ քայլից երբ մենք բացառեցինք օտար տողերը, մենք կիսում ենք մեր տվյալները ըստ դասերի, պայմանականորեն նշանակելով այդ 2 ենթամատրիցները հետևյալ կերպ՝ Xmin, Xmaj, համապատասխանաբար փոքրամասնության դասի և մեծամասնության դասին պատկանող տողերով։ Հաջորդիվ Min\_Max scaling-ի մեթոդով տվյալները նորմավորվում են, կրկին ըստ սյուների հետևյալ բանաձևով՝

ինչի արդյունքում տվյալները տեղափոխվում են 0-ից 1 միջակայք, հետագա հաշվարկները հեշտացնելու համար։

Այնուհետև պետք է կառուցվի հեռավորությունների մատրիցը տողերի նկատմամբ։ Հեռավորությունը հաշվարկվում է Էվկլիդյան նորմի բանաձևով՝

Հեռավորության մատրիցը կազմվում է եթե կազմվում է միայն օրինակ փոքրամասնության դասի համար, ապա որպես արդյունք ստանում ենք (nmin \* nmin-1) չափսի մատրից որտեղ բացառված են անկյունագծի տվյալները քանի որ դրանք 0-ներ են և ստացվեց որ մատրիցի aij էլեմենտի արժեքը ցույց է տալիս փոքրամասնության i-րդ տողը ինչ հեռավորություն ունի փոքրամասնության դասին պատկանող j-րդ տողից։ Հաջորդ մեծությունը, որը կօգտագործենք, Մոտ գտնվելու Գործոնն է/ Closeness Factor (C), որը հաշվարկվում է հետևյալ բանաձևով, բոլոր nmin-երի համար՝

որտեղ D(Xi) ստացվում է հետևյալ կերպ՝

Այսինքն՝ հեռավորության մատրիցի տողերի էլեմենտների գումարն է D վեկտորը, որտեղ ամեն էլեմենտ յուրաքանչյուր փոքրամասնության տողի համար գումարային հեռավորությունն է ցույց տալիս մնացած տողերի նկատմամբ։

Ալգորիթմին տրված պարամետրերից մյուսն էլ հեռավորության մատրիցից ստացվող ամենամոտ հարևանների քանակն է k-ն, Overlapping-ի խնդիրը լուծելու համար անհրաժեշտ է գտնել ամեն փոքրամասնության տողին համապատասխանող k ամենամոտ փոքրամասնության և k հատ ամենամոտ մեծամասնության դասին պատկանող տողերը։ Ինչից ստացվում է հետևյալ բազմությունները, յուրաքանչյուր i-րդ փոքրամասնության տողի համար հավաքագրվում է հետևալ բազմությունները {Smin( i, 1), … , S\_min( i, k)}, և {Smaj( i, 1), …, Smaj(i, 1)}, երբ i = 1, …, nmin

Այնուհետև հաշվարկվում է յուրաքանչյուր փոքրամասնության տողի համար τmin, τmaj, որոնք հաշվարկվում են հետևյալ բանաձևերով՝

τmin, τmaj, հաշվարկելու համար բանաձևի մեջ տեղադրվում է համապատասխանաբար Smin, Smaj, բազմությունները։

Տաոները ստացվում են nmin չափանի վեկտորներ և Overlapping-ի խնդիրը լուծելու համար որպեսզի բացառենք մեծամասնության դասից ավելի շատ հարևան ունեցող փոքրամասնության տողերը, կիրառում ենք հետևյալ տրամաբանությունը՝

i = 1, …, nmin

եթե այս պայմանը բավարարվում է ապա այդ տողը պահվում է:

1. Տողերի բաշխումը

Նախորդ քայլից հետո փաստացի մնացին այն տողերը որոնք outlier չեն և overlapping-ի խնդրիր չեն առաջացնում։ Այնուհետև հաշվարկվում է հետևյալ մեծությունը՝

φ նույնպես վեկտոր է, որի հաշվարկման ժամանակ հայտարարի յուրաքանչյուր էլեմենտին գումարվում է մեկ որպեսզի բացառվի 0-ի վրա բաժանումը։ Այնուհետև նորմավորելով վեկտորը ստանում յուրաքանչյուր տողի հավանականային վեկտորը հետևյալ կերպ՝

Այսպիսով ստացվեց փոքրամասնության դասին պատկանող մնացած տողերի հավանականային վեկտորը։ Հաջորդ քայլում արդեն կդիտարկենք թե ինչպես են նոր տողերը գեներացվում։

1. Նոր տողերի գեներացում

Այս քայլում հարկավոր է տալ ալգորիթմի վերջին պարամետրը, որը հարևանների քանակն է (թե քանի հատ մոտակա հարևանների օգնությամբ է նոր տողը գեներացվելու) նշանակենք դա m-ով։ Նախորդ քայլից հետո մնացած տվյալների համար հարկավոր է գտնել ամեն տողի համար իր ամենամոտ m հատ հարևաններին։ Նոր տողը գեներացվում է հետևյալ կերպ, ընտրվում է կամայական մնացած տողերից մեկը, վերցվում է այդ տողին ամենամոտ m հատ տողերը, վերցնում ենք այդ տողերին համապատասխանող P արժեքները, նոր տողը ստացվում է հետևյալ կերպ՝

Որտեղ α(j) 0-ից 1-ը ընկած պատահական մեծություն է, այսինքն ստացվում է որ վերցված տողերի էլեմենտները բազմապատկվում են իրենց հավանականային արժեքներով ու որոշակի պատահական մեծություննով և վերջում յուրաքանչյուր սյան էլեմենտներ իրար են գումարվում։ Ընդ որում կարելի է ուղղակի α(j) \* P(j) ստանալ օժանդակ ֆունկցիայի միջոցով նորմավորված վեկտոր, այսինքն նոր տողը ստանալ հետևյալ կերպ՝

Որտեղ (j) ստացվում է հետևյալ կերպ՝

Հետևյալ մեթոդով գեներացնելով նոր տողեր մենք նաև բացառում ենք overfitting-ի խնդիրը, այսինքն՝ պարզապես փոքրամասնության դասին պատկանող կետին մոտ պատահական կերպով նոր տող չենք գեներացնում։

Եզրակացություն․

Նախքան բալանսի բերելը տվյալները, հիմնականում Ալգորիթմները բավականին ցածր արդյունք էին ստանում Recall score, համեմատած Precision-ի։ Բալանսավորելու հիմնական նպատակը հենց այն էր որ ավելի բարձրացնենք Recall-ը, որպեսզի ավելի բարձր վերջնական F1 score ստանանք։ Բալանսավորման արդյունքում ստացվեց բարելավել միջինում մոտ 10%-ային կետով Recall, իսկ precision-ը գրեթե 20%-ով։ Արդյունքում մոտ 15%-ային կետով բարելավվեց F1 score-ը։

Այնուամենայնիվ վիզուալիզացիայի արդյունքում պարզ է որ թերություններ, մասնավորապես, դա ակնհայտ է այն փոփոխականների համար, որոնք որ ընդունում են վերջավոր թվով հնարավոր արժեքներ՝ դիսկրետ են (կատեգորիկ են)։ Կարելի է փորձել տրամաբանություն ավելացնել ալգորիթմի մեջ, այնպես որ այն կատեգորիկ սյուների համար վերջնական գեներացված տողերի այն սյուները որոնք որ կատեգորիկալ են, արժեքները կլորացնի։ Նաև կարելի է փորձել մոդելները բարելավել տարբեր ձև՝ Recall score մեծացնելու համար։

1. Նշված ալգորիթմը հիմնված է [հետևյալ](https://sci-hub.do/10.1109/ACCESS.2020.3003346) հոդվածի վրա [↑](#footnote-ref-1)