Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Мастер академске студије

Модул: Информатика у инжењерству

Предмет: Вештачка интелигенција

Број индекса: 396/2018

Никола Р. Петровић

Вишеслојни перцептрон у класификацији изразито небалансираног скупа података

Мастер рад

Комисија за преглед и одбрану:

1. **Проф. др Весна Ранковић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру овог мастер рада кандидат треба да:

1. Имплементира вишеслојна неуронска мрежа која ће вршити класификацију небалансираног скупа података
2. Имплементира алгоритам за балансирање скупа података за обучавање
3. Предложену структуру да употреби за класификацију узорака са различитим степеном дисбаланса класа

Препоручена литература:

[1] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, Jul. 2016, pp. 4368–4374, doi: 10.1109/IJCNN.2016.7727770.

[2] F. Chollet, *Deep learning with Python*. Shelter Island, New York: Manning Publications Co, 2018.

[3] J. Patterson and A. Gibson, *Deep learning: a practitioner's approach*, First edition. Sebastopol, CA: O'Reilly, 2017.

Крагујевац, 9.3.2020

Ментор:

Проф. др Весна Ранковић

_____

Садржај

Abstract

The paper is divided into several parts, the first part shows the description of the work of neural networks as the basis of artificial intelligence and the author considered that it is necessary to explain the way they function. The second part consists of a presentation of various solutions used in solving the problem of unbalanced data set, a large part refers to the resampling of the minority class and the subsampling of the majority class. The last part of the paper presents a practical example of teaching a neural network to recognize payment card fraud.

Keywords: neuron, perceptron, learning

Резиме

Рад је подељен у неколико целина, прва целина приказује опис рада неуронских мрежа као основе вештачке интелигенције и аутор је сматрао да је потребно објаснити начин функционисања истих. Други део се састоји од приказа различитих решења која се користе у решавању проблема небалансираног скупа података, велики део се односи на преузорковање мањиске класе и подузорковање већинске класе. Последњи део рада приказује практичан пример учења неуронске мреже да препозна преваре са платним картицама.

Кључне речи: неурон, перцептрон, учење

1. Увод

У последњих неколико година, вештачка интелигенција (AI) је доста популаризована. Машинско учење, дубоко учење и AI се свуда помињу чак и у текстовима који нису првенствено везани за информатику и технологију. Вештачка интелигенција обећава будућност где је све на дохват руке, где је све аутоматизовано и решиво употребом исте, али са друге стране AI приказује и будућност у којој би дошло да губитка послова и можда чак потпуног преузимања контроле од стране вештачке интелигнеције. За будућег или тренутног стручњака за машинско учење важно је да буде у стању да препозна квалилтетне информације од свих оних који се приказују у различитим публикацијама. У питању је будућност човечанства у коме је потребно активно учествовати.

Учење са неуравнотеженим подацима односи се на сценарио у којем количине инстанци које представљају концепте у датом проблему прате другачију расподелу. Главно питање када се решава такав проблем учења је када се тачност постигнута за сваку класу такође разликује. До ове ситуације долази јер је процес учења већине алгоритама класификације често пристрасан према примерима већинске класе, тако да мањински нису добро моделирани у коначном систему. Будући да је врло чест сценарио у стварним апликацијама, интересовање истраживача и практичара за ову тему је порасло током ових година.

Недавно су брзи развој науке и технологије промовисали раст и доступност података експлозивном брзином у различитим доменима. Све већа количина података и све сложенија структура података воде у такозвану „еру великих података“. Ово доноси сјајну прилику за истраживање података и откривање знања, као и многе изазове. Важан изазов је неравнотежа података. Иако је све више и више сирових података лако доступно, од којих је већина неуравнотежено расподељена, тј. неколико је података у изобиљу, док други имају само ограничен број. Ово се назива проблемом „небаласираних класа“ у заједници прикупљања података и изражено је у скоро свим прикупљеним скуповима података [1]. На пример, у клиничким дијагностичким подацима, већина људи је здрава, док је само прилично мали број њих нездрав. У класификационим задацима скупови података се обично класификују у бинарне класе података и вишеразредне скупове података према броју класа. Сходно томе, класификација може бити категорисана као бинарна класификација и класификација више класа. За скуп података бинарне класе, скуп података се назива неуравнотеженим ако је мањинска класа недовољно заступљена у односу на већинску класу, нпр. Већинска класа озбиљно представља мањинску класу .

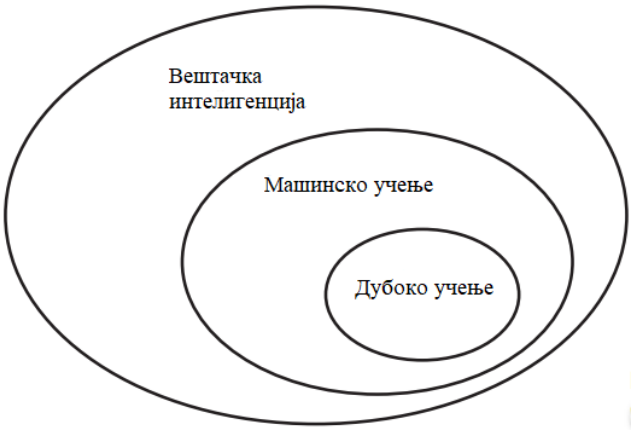
Неравнотежа података може довести до неочекиваних грешака, па чак и до озбиљних последица у анализи података, посебно у задацима класификације. То је зато што искривљена дистрибуција примера класе приморава алгоритме класификације да буду пристрасни већинској класи. Стога се концепти мањинске класе не уче на адекватан начин. Као резултат тога, стандардни класификатори (класификатори не узимају у обзир неравнотежу података) имају тенденцију да погрешно класификују мањинске узорке у већинске узорке када су подаци неуравнотежени, што резултира прилично лошим перформансама класификације. То може довести до високе цене у стварном животу. Узимајући у обзир горњи пример података о дијагнози, очигледно је да пацијент чини мањинску класу, док здраве особе чине већину. Ако се пацијенту погрешно дијагностикује као здрава особа, то би одложило најбоље време лечења и проузроковало значајне последице[1].

Међутим, на пољу дубоког учења по овом питању је обављен врло ограничен посао према најбољим сазнањима. Већина постојећих алгоритама дубоког учења не узима у обзир проблем неравнотеже података. Као резултат, ови алгоритми могу имати добру изведбу на уравнотеженим скуповима података, док се њихови учинци не могу зајамчити на неуравнотеженим скуповима података.

2. Вештачка интелигенција, машинско учење и дубоко учење

2.1 Подела

Прво је потребно дефинисати јасно о чему се говори када се помене вештачка интелигнеција (AI). Шта је то вештачка интелигенција, машинско учење и дубоко учење, расподела је приказана на слици 1 [2].

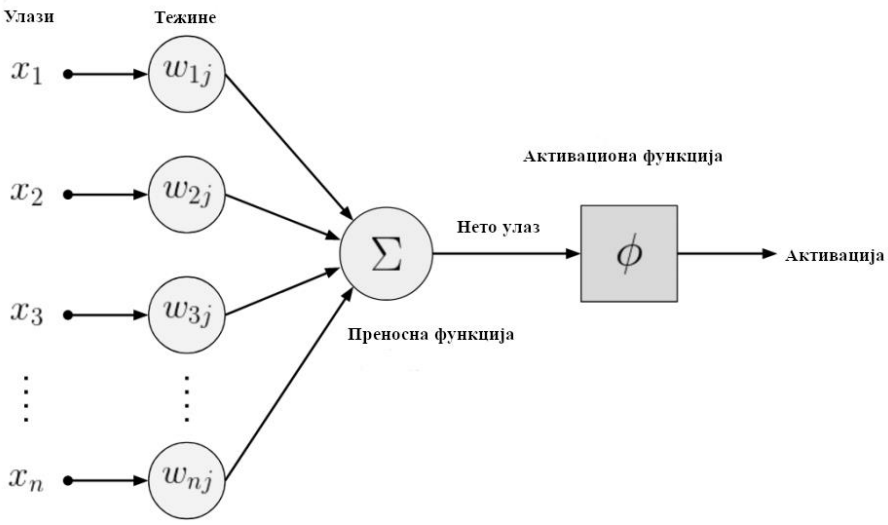


Слика 1 - Вештачка интелигенција, машинско учење и дубоко учење [2]

3. Вишеслојна неуронска мрежа

Вишеслојна мрежа је неуронска мрежа са улазним слојем, једним или више скривених слојева и излазним слојем. Сваки слој има један или више вештачких неурона. Ови вештачки неурони су слични њиховом прекурсору перцептрону, али имају различиту активациону функцију у зависности од посебне намене слоја у мрежи[3].

Вештачки неурон вишеслојног перцептрона сличан је свом претходнику, перцептрону, али додаје флексибилност у врсти активацијског слоја који се може користити. На слици (**Слика *2***) приказан је ажурирани дијаграм за вештачки неурон који је заснован на перцептрону.



Слика 2 - Вештачки неурон вишеслојног перцептрона [2]

Овај дијаграм је сличан за једнослојни перцептрон, али може се приметити генерализована функцију активирања.

Тежине веза. Тежине на везама у неуронској мрежи су коефицијенти који скалирају (појачавају или минимизују) улазни сигнал за одређени неурон у мрежи. У уобичајеним приказима неуронских мрежа, то су линије / стрелице које иду од једне тачке до друге, ивице математичког графикона. У математичким приказима неуронских мрежа везе се често означавају као w.

Прагови активације су скаларне вредности додате на улазу да би се осигурало активирање бар неколико чворова по слоју без обзира на јачину сигнала. Прагови активације допуштају да се догоди учење у случају слабог сигнала. Омогућавају мрежи да испроба нове интерпретације или понашања. Прагови активације су обично означени као b, и, као и тежине се мењају током процеса учења.

Активационе функције. Функције које управљају понашањем вештачког неурона називају се активацијским функцијама. Пренос тог улаза познат је као пропагација у напред. Функције активирања трансформишу комбинацију улаза, тежине и прагове активације. Производи ових трансформација уносе се за следећи слој чвора. Многе (али не све) нелинеарне трансформације које се користе у неуронским мрежама трансформишу податке у погодан распон, као што је 0 до 1 или -1 до 1. Када вештачки неурон прелази на не-нуларну вредност на други вештачки неурон, каже се да је активиран.

4. Основне поставке о неуроноским мрежама

Добро обучена вештачка неуронска мрежа има тежине који појачавају сигнал и пригушују неправилне податке. Већа тежина значи чвршћу повезаност сигнала и резултата мреже. Улази упарени с великим тежинама ће утицати на интерпретацију података мреже више него улази упарене с мањом тежином [4].

Процес учења за било који алгоритам учења помоћу тежина је процес прилагођавања тежина и прагова активације, чинећи једне мање а друге веће, при чему се додељује значај одређеним деловима информација и минимизују други делови. Ово помаже моделу да сазна

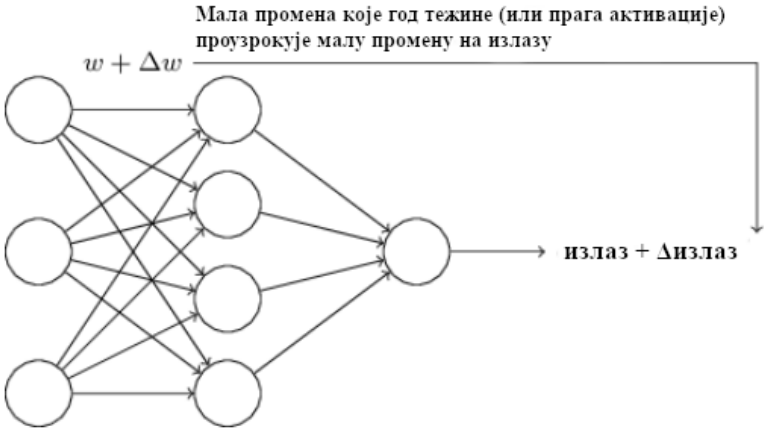
које карактеристике су везана за које исходе и прилагођава тежине и прагова активације према томе [4].

У већини скупова података одређене карактеристике су чврсто повезане са одређеним ознакама (нпр., Квадратура се односи на продајну цену куће). Неуронске мреже слепо уче о тим односима тако што погађају на основу улаза и тежине, а затим мере колико су тачни резултати. Функције губитака у алгоритам оптимизације, као што је стохастички пад градијента (SGD), награђују мрежу за добра нагађања и кажњавају је за лоше. SGD помера параметре мреже према прављењу добрих предвиђања и даље од лоших[4].

Други начин да се посматра процес учења је да се на етикете гледа као на теорије и да се карктерситике посматрају као доказ. Затим је могуће направити аналогију којом мрежа настоји успоставити повезаност између теорије и доказа. Модел покушава да одговори на питање „коју теорију подржавају докази?“

4.1 Неурони који користе сигмоидну функцију

Питање је како се може осмислити алгоритам за неуронску мрежу? Претпоставља се да постоји мрежа перцептрона коју је потребно искористити како ви се решио неки проблем. На пример, улази у мрежу могу бити необрађени подаци пиксела са скениране, руком исписане слике цифара. И потребно је да мрежа научи тежине и прагове активације тако да излаз из мреже правилно класификује цифру. Уколико је направљена мала промена неке тежине (или прага активације) у мрежи. Претпоставка је да ова мала промена у тежини узрокује само малу одговарајућу промену у излазу из мреже. Ово омогућава учење могућим. Илустрација (**Слика *3***) приказује неуронску мрежу која би се користила у ту сврху[2].

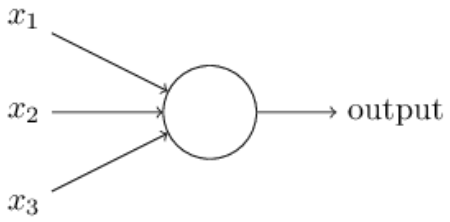


Слика 3 - Промена парамтера неуронске мреже[2]

Ако је тачно да мала промена у тежини (или прагу активације) изазива само малу промену у излазу, онда би се могла искористити ова чињеница како би се измениле тежине и прагови активације да би се мрежа понашала више на начин који је пожељан. На пример, мрежа грешком класификовала слику као „8“ када би она требала бити „9“. Онда би било могуће направити малу промену у тежинама и праг. активације тако да се мрежа мало приближи класификацији слике као „9“. Потом би се поступак поновио, мењајући тежине и прагове активација изнова и изнова како би се постигао све бољи и бољи резултат. Мрежа би учила[2].

Проблем је у томе што се то не догађа када мрежа садржи перцептроне. У ствари, мала промена у тежини или прага активацвиције било ког појединог перцептрона у мрежи понекад може узроковати да се излаз тог перцептрона у потпуности буде погрешан, рецимо од 0 до 1. Тај преокрет може потом да проузрокује понашање остатка мреже да се промени на врло компликован начин. Иако је број „9“ сада можда исправно класификован, понашање мреже на свим осталим сликама ће се вероватно потпуно променити на неки тешко контролисан начин. Због тога је тешко видети како постепено модификовати тежине и пристраности тако да се мрежа приближи жељеном понашању. Можда постоји неки паметан начин за решавање овог проблема. Али није одмах очигледно како је могуће добити мрежу перцептрона за учење.

Овај проблем се може превазићи увођењем нове врсте вештачког неурона зване сигмоидни неурон (**Слика *4***). Сигмоидни неурони су слични перцептронима, али модификовани тако да мале промене у тежини и пристраности узрокују само малу промену у њиховом излазу. То је кључна чињеница која ће мрежи сигмоидних неурона омогућити да учи.



Слика 4 - Сигмоидни неурон[2]

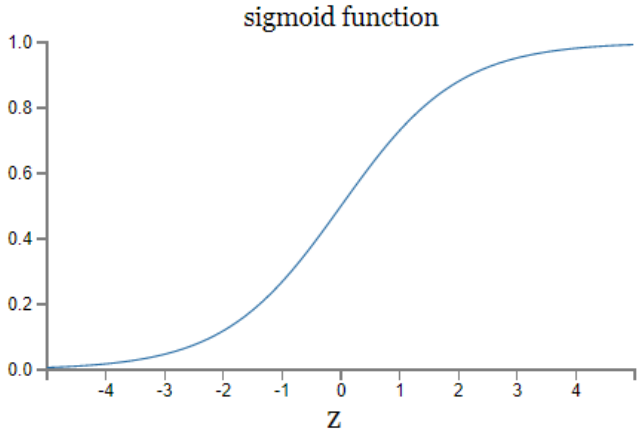
Баш као и перцептрон, сигмоидни неурон има улазе, $x_1, x_2, \ldots$ Али уместо да буду само 0 или 1, ови улази могу да приме и било које вредности између 0 и 1. Дакле, на пример, 0.638… је валидан улаз за сигмоидни неурон. Такође као перцептрон, сигмоидни неурон има тежину за сваки улаз $w_1, w_2, \ldots$,и укупан праг активације, b. Али излаз није 0 или 1. Уместо тога, то је $\sigma(w \cdot x + b)$, где се σ назива сигмоидна функција. А дефинише се:[5]

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}} \qquad 4.1$$

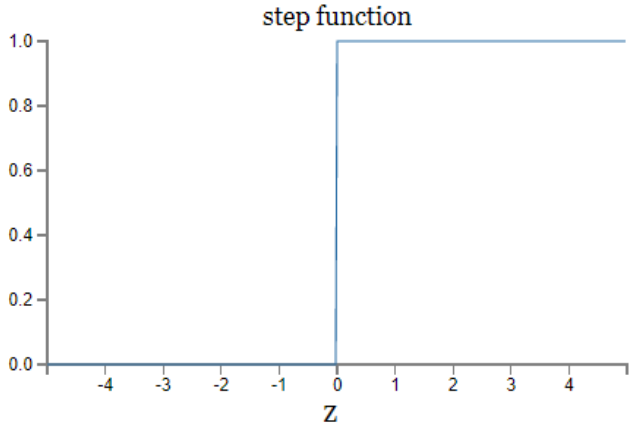
Како би се све мало експлицитније приказало, излаз сигмоидног неурона са улазима $x_1, x_2, \ldots$, тежинама $w_1, w_2, \ldots$ и праговима активације b је:

$$\frac{1}{1 + \exp\left(-\sum_j w_j x_j - b\right)} \qquad 4.2$$

На први поглед сигмоидни неурони изгледају врло различито од перцептрона. Алгебарски облик сигмоидне функције може изгледати помало чудно, постоји много сличности између перцептрона и сигмоидних неурона. Сигмоидна функција (**Слика *5***)[2].



Слика 5 - Сигмоидна гункција[3]



Слика 6 - Степ функција[3]

Ако је σ у ствари степ функција, тада би сигмоидни неурон био перцептрон, јер би излаз био 1 или 0 у зависности да ли је $w \cdot x + b$ позитиван или негативан. Заправо, када је в·к + б = 0 перцептрон излази 0, док функција корака даје 1. Употребом стварне функције σ добија се, као што се већ подразумева горе, заравњен перцептрон. Заправо, пресудна је чињеница заравњења функције σ, а не њен детаљан облик. Глаткост σ значи да ће мале промене Δw_j у тежинама и Δb у праговима активације произвести малу промену Δизлаз у излазу из неурона. У ствари, рачун говори да је Δизлаз добро апроксимиран према:[6]

$$\Delta\text{излаз} \approx \sum_j \frac{\partial\text{излаз}}{\partial w_j}\Delta w_j + \frac{\partial\text{излаз}}{\partial b}\Delta b \qquad 4.3$$

где је збир преко свих тежина, w_j и ∂излаз/ ∂w_j и ∂излаз/ ∂b означавају парцијалне изводе излаза у односу на w_j и b.

Како се тумачи излаз из сигмоидног неурона? Очигледно, једна велика разлика између перцептрона и сигмоидних неурона је да сигмоидни неурони немају излазе 0 или 1. Они

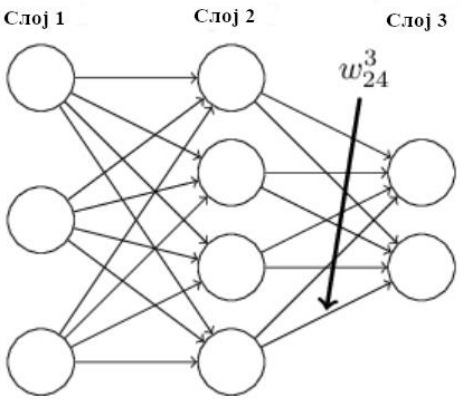
могу имати као излаз било који стварни број између 0 и 1, тако да су вредности попут 0,173… и 0,689… легитимни излази. Ово може бити корисно, на пример, ако је потребно користити излазну вредност која представља просечни интензитет пиксела на улазу слике у неуронску мрежу. Али понекад то може бити сметња. Уколико је потребно да излаз из мреже показује "улазна слика је 9" или "улазна слика није 9". Очигледно, то би било најлакше учинити ако је излаз 0 или 1, као у перцептрону. Али у пракси се може успоставити конвенција која ће се бавити тиме, на пример, одлуком да било који излаз од најмање 0,5 протумачи као назнаку „9“, а било који мањи од 0,5, што означава „није 9“[2].

5. Алгоритам са пропагацијом грешке уназад

Алгоритам са пропагацијом грешке уназад првобитно је уведен 1970-их, али његова важност није у потпуности уважена све до чувеног рада Дејвида Румелхарта, Џофрија Хинтона и Роналда Вилијамса (енг. David Rumelhart, Geoffrey Hinton, and Ronald Williams) из 1986. године. У овом раду је описано неколико неуронских мрежа где пропагација грешке уназад делује много брже него ранији приступи учењу, омогућавајући употребу неуронских мрежа за решавање проблема који су претходно били нерешиви. Данас је алгоритам са пропагацијом грешке уназад незаобилазан код учења у неуронским мрежама[5].

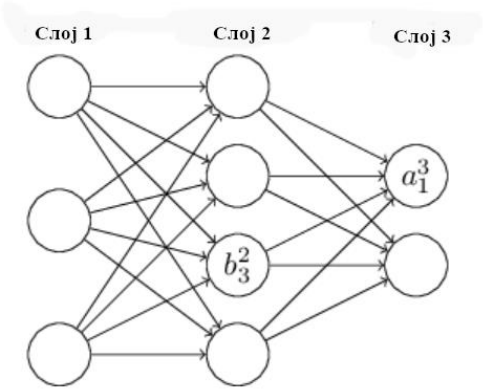
У срцу алгоритма са пропагацијом грешке уназад налази се израз парцијалног извода $\partial C / \partial w$ критеријумске функције C у односу на било коју тежину w (или праг активације b) у мрежи. Израз говори како се брзо мења критеријумска функција када се мењају тежине и прагови активације[5].

Користиће се w_{jk}^{l} како би се означила тежина за везу од k – тог неурона у $(l-1)$ слоју и j - тог неурона у 1. слоју. Тако, на пример, доњи дијаграм (**Слика *7*** - Пример означавања тежина) приказује тежину везе од четвртог неурона у другом слоју до другог неурона у трећем слоју мреже:[7]



Слика 7 - Пример означавања тежина[3]

Слична нотација се користи за означавање прагова активације и самих активација неурона. Користи се b_j^l за праг активације j-ог неурона у l -том слоју. А a_j^l за активацију j-ог неурона у l слоју. Следећи дијаграм приказује примере ових употреба:[5]



Слика 8 - Пример означавања прагова активације и активација неурона[3]

Са овим нотацијама, a_j^l активације j-ог неурона у l - том слоју повезан је са активацијама у $(l-1)$ слоју једначином:[5]

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \qquad 5.1$$

где је сума свих неурона k у $(l-1)$ слоју. Да би се поново написао овај израз у облику матрице, дефинише се матрица тежине w^l за сваки слој, l. Уноси матрице тежине су само тежине које се повезују са 1. слојем неурона, то јест унос у j ред и k колону је w_{jk}^l. Слично томе, за сваки слој l дефинише се вектор прагова активцације, b^l. Према томе, компоненте вектора прагова активације су само вредности b_j^l, једна компонента за сваки неурон у l слоју. И на крају, дефинише се вектор активација a^l чије су компоненте активације a_j^l.

Последњи термин који је потребан како би се једначина 5.1 преписала у матрични облик је идеја векторизације функције као што је σ. Односно, компоненте $\sigma(v)$ су само $\sigma(v)_j = \sigma(v_j)$. Као пример, ако је дата функција $f(x) = x^2$, онда векторизовани облик f има ефекат:[5]

$$f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix}f(2)\\f(3)\end{bmatrix} = \begin{bmatrix}4\\9\end{bmatrix} \qquad 5.2$$

Са овом нотацијом на уму, једначина 5.1 се може написати као:

$$a^l = \sigma(w^l a^{l-1} + b^l) \qquad 5.3$$

Овај израз претставља глобалнији начин размишљања о томе како се активације у једном слоју односе на активације у претходном слоју: само је потребно применити матрицу тежине на активације, затим додати вектор прагова активације и на крају примени σ функција. Тај глобални поглед је често лакши и сажетији (и укључује мање индекса) поглед од неурона до неурону који је узет до сада. Израз је такође користан у пракси, јер већина

библиотека матрица пружа брзе начине имплементације множења матрице, додавања вектора и векторизације.

Када се користи једначина 5.3 како би се би израчунало a^l, израчунава се и величина $z^l \equiv w^l a^{l-1} + b^l$. Ова величина се показује довољно корисном да је вредна именовања: z^l се назива тежински улаз неурона у l слоју. Такође је вредно поменути да z^l има компоненте $z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$ односно, z^l је само тежински улаз у функцију активирања неурона j у слоју l[5].

5.1 Претпоставке око критеријумске функције

Циљ алгоритма са проп. грешке уназад је израчунати парцијалне изводе $\partial C / \partial w$ и $\partial C / \partial b$ критеријумске функције C у односу на било коју тежину w или праг активације b у мрежи. Да би проп. грешке уназад успела, потребни је направити две главне претпоставке о облику критеријумске функције. Пре него што се те претпоставке, корисно је имати на уму критеријумску функцију. Користи се квадратна функција трошкова, једначина.У нотацији која је претходно описана, функција има облик:[8]

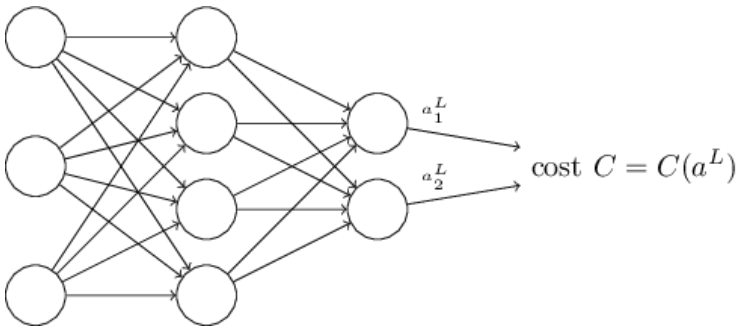
$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2 \qquad 5.4$$

где је n укупан број улаза за тренинг, сума је преко свих индивидуланих улаза за тренинг, x, y = y(x) је припадајући жељени излаз, а $a^L = a^L(x)$ је вектор активационих излаза из мреже онда када је x улаз.

Питање се поставља, које то претпоставке је потребно донети у вези C функције, како би се пропагација грешке уназад остварила. Прва претпоставка је та да се функција може написати као средња вредност $C = \frac{1}{n} \sum_x C_x$ критеријумске функције C_x за индивидуалне улазе приликом тренинга, x. Ово је случај за квадратну критеријумску функцију, где је цена за један илаз $C_x = \frac{1}{2} \|y - a^L\|^2$ [8].

Разлог зашто је потребна ова претпоставка је тај што оно што пропагација грешке уназад заправо допушта да се израчунају парцијални изводи $\partial C_x / \partial w$ и $\partial C_x / \partial b$ за један пример тренинга. Затим се враћа $\partial C / \partial w$ и $\partial C / \partial b$ рачунањем просека примера тренинга. У ствари, имајући на уму ову претпоставку, може се претпоставити да је улаз x фиксиран и потом потпуно одбацити суфикс x, тиме поистовећујући критеријум C_x као C.

Друга претпоставка о критеријумској функцији је та да се критеријум или цена могу записати као функција излаза из неуронске мреже (Слика 9)[8].



Слика 9 - Критеријум на излазу из мреже[2]

На пример, квадратна критеријумска функција задовољава овај захтев, јер квадратни трошак за један пример тренинга x може бити записан као[8]

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2 \qquad 5.5$$

и самим тим је ово функција излазних активација. Наравно, ова критеријумска функција такође зависи од жељеног резултата y.

5.2 Четри основне једначине алгоритма пропагације уназад

Алгоритам пропагације грешке уназад подразумева разумевање како промена тежина и прагова активација у мрежи мења функцију трошкова (критеријумску функцију). То значи рачунање парцијалних извода $\partial C / \partial w_{jk}^l$ и $\partial C / \partial b_j^l$. Али да би се израчунали, прво је потребно увести средњу количину, δ_j^l, која се назива грешком у j-том неурону у l – том слоју. Пропагације грешке уназад ће дати поступак за израчунавање грешке ,а затим ће се односити δ_j^l на $\partial C / \partial w_{jk}^l$ и $\partial C / \partial b_j^l$[5].

Како би се лакше објаснило како се грешка дефинише у неуронској мрежи, пожељно је замислити да постоји неко ко управља неуронском мрежом и подешава сваки неурон. Та замишљена особа је тренутно на неурону j у слоју l. Када улаз дође на неурон, та особа мења параметре тог неурона. Дода малу промену Δz_j^l на улазну тежину неурона, па неурон нема излаз $\sigma(z_j^l)$ већ $\sigma(z_j^l + \Delta z_j^l)$. Ова промена пролази кроз касније слојеве мреже и коначна промена на излазу се мења за вредност $\frac{\partial C}{\partial z_j^l}\Delta z_j^l$ [5].

Сада ова замишљена особа покушава да поправи тај коначан излаз, тј. покушава да пронађе вредност Δz_j^l која чини критеријумску фунцкију мањом. Уколико се претпостави да $\frac{\partial C}{\partial z_j^l}$ има велику вредност (позитивну или негативну). Ова особа може да смањи вредност драстично уколико изабере Δz_j^l које има супртотну вредност од $\frac{\partial C}{\partial z_j^l}$. Уколико је $\frac{\partial C}{\partial z_j^l}$ близу нуле, онда та особа не може да промени коначну вредност мењајући вредност улаза z_j^l . Та са особа би онда утврдила да је неурон већ прилично оптималан. Па према томе може се закључити да је $\frac{\partial C}{\partial z_j^l}$ мера грешке у неурону[5].

Према овој причи, грешка δ_j^l неурона j у слоју l је дефинисана према:

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \qquad 5.6$$

Према досадашњим конвенцијама, користи се δ^l како би се означио вектор грешака повезаних са слојем l. Пропагација грешке уназад ће дати начин израчунавања δ^l за сваки слој, а затим повезивање тих грешака са количинама од стварног интереса, $\partial C / \partial w_{jk}^l$ и $\partial C / \partial b_j^l$[5].

Алгоритам пропагације грешке уназад заснован је на четири основне једначине. Заједно, те једначине дају начин израчунавања грешке δ^l и градијента критеријумске функције.

Једначина за грешку у излазном слоју, δ^L: Компоненте δ^L су дате са:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \qquad 5.7$$

Први израз на десној страни, $\frac{\partial C}{\partial a_j^L}$, управо мери колико се брзо трошак мења као функција активирања j- тог излаза. Ако, на пример, C не зависи много од одређеног излазног неурона, j, тада ће δ_j^L бити мали, што би се и очекивало. Други израз с десне стране, $\sigma'(z_j^L)$, мери колико се брзо мења функција активирања σ на z_j^L[2].

Приметно је да је све у јед 5.7 лако израчунати. Конкретно, z_j^L се рачуна док се рачуна понашање мреже и то је само мала додатна цена за израчунавање $\sigma'(z_j^L)$. Тачан облик $\partial C / \partial a_j^L$, наравно, зависи од облика критеријумске функције. Међутим, под условом да је позната критеријумска функција, не би требало да буде проблема са рачунањем $\partial C / \partial a_j^L$. На пример, ако се користи квадратна функција трошкова, тада је $C = \frac{1}{2}\sum_j (y_j - a_j^L)^2$, и тако је $\partial C / \partial a_j^L = (a_j^L - y_j)$, што је очигледно лако израчунати[2].

Једначина 5.7 је компонентни израз за δ^L. То је савршено добар израз, али не и матрични иоблик који је пожељан за алгоритам пропагације грешке уназад. Међутим, лако је поново написати једначину у облику заснованом на матрици, као[2]:

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad 5.8$$

Овде је $\nabla_a C$ дефинисан као вектор чије су компоненте парцијални изводи $\partial C / \partial a_j^L$. Може се мислити на $\nabla_a C$ као на израз брзине промене C у односу на излазне активације. Лако је видети да су једначине 5.7 и 5.8 еквивалентне, и због тога ће се од сада па надаље 5.7 наизменично користити за обе једначине. Као пример, у случају квадратне критеријумске фукниције је дефинисано $\nabla_a C = (a^L - y)$ тако да облик према 5.7 заснован на матрици постаје $\delta^L = (a^L - y) \odot \sigma'(z^L)$[2]

Једначина за грешку δ^l у смислу грешке у следећем слоју, δ^{l+1}: Конкретно:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \qquad 5.9$$

Ово помера грешку уназад кроз функцију активирања у слоју l, дајући грешку δ^l у тежинском улазу у слој l[5].

Комбиновањем 5.7 са 5.9 може се израчунати грешка δ^l за било који слој у мрежи. Почиње се са 5.7 за израчунавање δ^L, затим се примењује једначина 5.9 за израчунавање δ^{L-1}, затим једначина 5.9 поново за израчунавање δ^{L-2} и тако даље, све тако назад кроз мрежу.

Једначина за стопу промене трошкова (критеријума) у односу на било који праг активације у мрежи: Конкретно[5]:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \qquad 5.10$$

Односно, грешка δ_j^l тачно је једнака брзини промене $\partial C / \partial b_j^l$. Ово је веома погодно, јер преко 5.7 и 5.9 је већ израчунато δ_j^l. Може се преписати 5.10 у скраћеном облику као[5]:

$$\frac{\partial C}{\partial b} = \delta \qquad 5.11$$

где се подразумева да се δ процењује на истом неурону као и праг активације b.

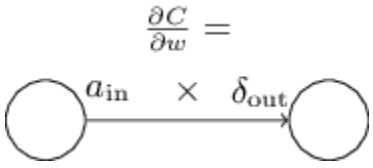
Једначина за стопу промене трошкова тј. критерујема у односу за било коју тежину у мрежи: Конкретно:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \qquad 5.12$$

Ово говори како израчунати парцијалне изводе $\partial C / \partial w_{jk}^l$ у смислу величина δ^l и a^{l-1}, за које већ познато како израчунати. Једначина се може преписати у јаснијој нотацији као:

$$\frac{\partial C}{\partial w} = a_{\text{in}} \delta_{\text{out}} \qquad 5.13$$

где се подразумева да је a_{in} активација неурона на улазу у тежину w, а δ_{out} је грешка у излазу неурона из тежине w. Ако се зумира и посматра само тежина w (Слика 10), и два неурона повезана том тежином, то се може приказати као[5]:



Слика 10 - Увећана веза између два неурона

Последица једначине једначине 5.13 је да ће, када је активација неурона мала, $a_{\text{in}} \approx 0$, градијентни израз $\partial C / \partial w$ такође бити мали. У овом случају, може се рећи да тежина учи полако, што значи да се током спуштања градијент не мења много. Другим речима, једна последица 5.12 је да се тежине који излазе из неурона са ниском активацијом полако уче[5].

Постоје и други закључци који се могу добити из једначина (5.8, 5.9, 5.10, 5.12). Уколико се посматра излазни слој. Примтиће се термин $\sigma'(z_j^L)$ у 5.8. Из графа сигмоидне функције се примећује да σ функција постаје равна када је $\sigma(z_j^L)$ приближно 0 или 1. Када се то догоди, онда је $\sigma'(z_j^L) \approx 0$. Дакле, закључује се да ће тежина у завршном слоју полако учити ако је излазни неурон или ниске активације (≈0) или високе активације (≈1). У овом случају је уобичајено рећи да је излазни неурон засићен и као резултат тежина је престала да учи (или полако учи). Сличне примедбе се односе и на прагове активације излазног неурона[5].

Сличан закључак се може добити и за раније слојеве. Конкретно, израз $\sigma'(z^l)$ у 5.9. То значи да ће δ_j^l вероватно бити мале вредности ако је неурон близу засићења. А то заузврат значи да ће све тежине унесене у засићени неурон учити споро.

Закључак је да ће тежина споро учити ако је или улазни неурон слабе активације, или ако је излазни неурон засићен, тј. или је висока или ниска активација.

Ниједно од ових запажања није превише изненађујуће. Ипак, помажу у побољшању менталног модела онога што се дешава док неуронска мрежа учи. Испоставило се да четири основне једначине важе за било коју функцију активације, а не само за стандардну сигмоидну функцију. Тако да ове једначине се могу користити за дизајнирање активационих функција које имају посебно жељена својства учења. Као, претпоставимо да је изабрана (не-сигмоидну) функцију активирања σ тако да је σ ′ увек позитивно и да се никада не приближи нули. То би спречило успоравање учења до којег долази када се обични сигмоидни неурони засићују.

Преглед онсновних једначина алгортма за пропагацију грешке уназад:

	$\delta^L = \nabla_a C \odot \sigma'(z^L)$	*Основна јед. 1*
	$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$	*Основна јед. 2*
	$\frac{\partial C}{\partial b_j^l} = \delta_j^l$	*Основна јед. 3*
	$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$	*Основна јед. 4*

5.3 Алгоритам за пропагацију грешке уназад

Једначине алгортима са пропагацијом грешке уназад пружају начин израчунавања градијента критеријумске функције функције. Изричито написано у облику алгоритма:

1. **Улаз $\boldsymbol{x}$:** Подесити одговарајућу активацију a^1 за улазни слој.
2. **Пропагација напред:** За свако $l = 2,3, \dots, L$ израчунати $z^l = w^l a^{l-1} + b^l$ и $a^l = \sigma(z^l)$
3. **Грешка на излазу $\boldsymbol{\delta^L}$:** Израчунати вектор $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. **Пропагација уназад:** За свако $l = L-1, L-2, \dots, 2$ израчунати $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. **Излаз:** Градијент критеријумске функције је дат према: $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ и $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

Испитујући алгоритам може се видети зашто се то назива повратно ширење. Рачунају се вектору грешке δ^l уназад, почевши од завршног слоја. Кретање уназад последица је чињенице да је критеријумска функција излаз из мреже. Да би се разумело како цена варира са ранијим тежинама и праговима активација, потребно је више пута применити ово правило ланца, радећи уназад кроз слојеве како би се добили употребљиви изрази[8].

6. Практична примена вишеслојног перцептрона на проблему класификације изразито небалансираног скупа података

Проблем класификације изразито небалансираног скупа је пример проблема класификације где је расподела примера у познатим класама пристрасна. Расподела може варирати од мале пристрасности до озбиљне неравнотеже гдје постоји један пример у мањинској класи, а стотине, хиљаде или милионе примера у већинској класи или класама.

Класификације изразито небалансираног скупа представљају изазов за предиктивно моделирање, јер је већина алгоритама машинског учења коришћених за класификацију дизајнирана око претпоставке једнаког броја примера за сваку класу. То резултира моделима са лошим предвиђањем, посебно за мањинску класу. Ово је проблем јер је типично мањинска класа важнија и самим тим је проблем осетљивији на грешке у класификацији мањинске класе од већинске класе.

6.1 Класификационо предиктивно моделирање

Класификација је проблем предиктивног моделирања који укључује додељивање ознаке класе сваком посматрању.

Модели класификације генеришу класу која долази у облику дискретне категорије. За већину практичних примена потребно је дискретно предвиђање категорија да би се донела одлука. [9]

Сваки пример се састоји од запажања и ознаке класе.

Пример: Запажање из домена (улаз) и придружена ознака класе (излаз).

На пример, могу се прикупити мере цвета и класификовати врсте цветова (ознаке) на основу мерења. Број класа за проблем предиктивног моделирања обично је фиксиран када је проблем дефинисан, а обично се број класа не мења.

Може се одлучити за предвиђање вероватноће чланства у класи уместо специфичне ознаке класе.

Ово омогућава предиктивном моделу да подели несигурност у предвиђању у низу опција и омогући кориснику да интерпретира резултат у контексту проблема.

Попут регресионих модела, и класификацијски модели производе континуирано вредновано предвиђање, које је обично у облику вероватноће (тј. Предвиђене вредности чланства у класи за било који појединачни узорак су између 0 и 1 и суме до 1)[9].

На пример, датим мерењима цвета (посматрање), може се предвидети вероватноћа да је цвет пример сваке од двадесет различитих врста цвета.

Број класа за проблем предиктивног моделирања обично је фиксиран када је проблем дефинисан, а обично се број класа не мења.

Проблем предиктивног моделирања класификације може имати две ознаке класа. Ово је најједноставнији тип класификационог проблема и назива се двокласна класификација или бинарна класификација. Наизменично, проблем може имати више од две класе, као што су три, 10 или чак стотине класа. Ове врсте проблема називају се вишекласним проблемима класификације.

- **Бинарни проблем класификације:** Проблем предиктивног моделирања класификације где сви примери припадају једној од две класе.
- **Проблем вишекласне класификације**: Проблем предиктивног моделирања класификације, где сви примери припадају једној од три класе.

Када се ради на проблемима класификовања предиктивног моделирања, мора се прикупити скупо података о обуци.

Скуп података о обуци је низ примера из домена који укључују и улазне податке (нпр. Мерења) и излазне податке (нпр. Ознаку класе).

Скуп података о обуци: Бројни примери прикупљени из домена проблема који укључују улазна запажања и ознаке класе излаза.

У зависности од сложености проблема и врста модела који се могу одабрати, можда ће бити потревне десетине, стотине, хиљаде или чак милиони примера из домена како би се направио скуп података о обуци.

Скуп података за обуку користи се за боље разумевање улазних података како би се најбоље припремило за моделирање. Такође се користи за процену скупа различитих алгоритама за моделирање. Користи се за подешавање хиперпараметара изабраног модела. И на крају, скуп података за обуку користи се за обуку коначног модела на свим доступним подацима који се могу користити у будућности за предвиђање нових примера из домене проблема.

6.2 Проблеми са неуравнотеженом класификацијом

Број примера који припадају свакој класи може се назвати дистрибуцијом класе.

Неуравнотежена класификација односи се на проблем предиктивног моделирања класификације где број примера у скупу података за обуку за сваку ознаку из скупа није уравнотежен.

Односно, тамо где расподела класа није једнака или близу једнаке, већ је пристрасна или „искривљена“ (енг. skewed).

Неуравнотежена класификација: Проблем предиктивног моделирања класификације где расподела примера у класама није једнака.

На пример, могу се сакупити мере цвећа и то 80 примера једне цветне врсте и 20 примера друге цветне врсте, а само ови примери чине скуп података за обуку. Ово представља пример неуравнотежене класификације.

Неуравнотеженост се дешава када једна или више класа има врло мале пропорције у подацима о обуци у поређењу са осталим часовима[9].

Постоје и друга мање општа имена која се могу користити за описивање ових врста проблема класификације, као што су:

- Предвиђање ретких догађаја.
- Предвиђање екстремних догађаја.
- Тешка класна неравнотежа.
- Изразито небалансирани скуп података

Уобичајено је описивати неравнотежу класа у скупу података кроз однос.

Дисбаланс класе мора бити дефинисан у односу на одређени скуп података или дистрибуцију. С обзиром да су ознаке класе потребне да би се утврдио степен класне неравнотеже, класна неравнотежа се обично мери с обзиром на расподелу обуке.[10]

На пример, неуравнотежени бинарни проблем класификације са неравнотежом од 1 према 100 (1: 100) значи да за сваки пример у једној класи постоји 100 примера у другој класи.

Други начин за описивање дисбаланса класа у скупу података је сумирање дистрибуције класа у процентима од скупа података о обуци. На пример, неуравнотежени проблем класификације више класа може имати 80 процената примера у првој класи, 18 процената у другој класи и 2 процента у трећој класи.

6.3 Приказ проблема класификације

У бинарно небалансираним скуповима података, број случајева једне класе је већи од броја друге класе. Због тога је прва класа позната као већинска, а друга класа као мањинска. Стога овај скуп података садржи две врсте случајева: већину и мањину. Дистрибуција случајева у неуравнотеженим бинарним скуповима података мери се дисбалансираним односом (IR)[11] који је дефинисан у једначини:

$$IR = \frac{Broj\ instanca\ u\ vecinskoj\ klasi}{Broj\ instanca\ u\ manjinskoj\ klasi} \qquad 6.1$$

Према вредности IR, неуравнотежени скупови података подељени су у три класе [12]: скупови података са малим дисбалансом (IR је између 1,5 и 3), скупови података са средњим дисбалансом (IR је између 3 и 9) и скупови података са великим дисбалансом (IR је већи од 9).

6.4 Домени примене

Неуравнотежени скупови података појављују се у следећих неколико домена.

1. **Процена ризика**

Сваке године телекомуникациона индустрија трпи милијарде долара ненаплативих дугова. Стога је ненаплатива контрола главни проблем у индустрији. Једно од решења је

коришћење велике количине историјских података за изградњу модела који се користе за процену ризика за сваког клијента клијента или за сваку трансакцију као подршку управљању ризицима који смањује ниво ненадокнадивог дуга. Међутим, у скупу података неплаћање купаца укључује неколико процената становништва [13].

2. **Постављање дијагнозе у медицини**

Клинички скупови података чувају велике количине података о пацијенту. На овим скуповима података примењује се техника рударења података како би се открили односи и трендови између клиничких и патолошких података. Њен циљ је да разуме еволуцију и карактеристике одређених болести. Међутим, у овим скуповима података случајеви болести су ређи међу нормалним становништвом [14].

3. **Откривање упада у мреже**

Мрежни рачунарски системи све више играју виталну улогу у модерним друштвима. Напади на рачунарске системе и мреже расту. Постоје различите категорије мрежних напада; неке су бројне, а друге ретке. На пример, скуп података KDD-CUP'99 садржи четири категорије мрежних напада: ускраћивање услуге (Denial of service(DoS)), надгледање (probe), (root to local (R2L)) и (user to root (U2R)). Последња два напада су суштински ретка [15].

6.5 Метрика евалуације

Класичне мере перформанси које се користе за оцењивање перформанси класификатора када се користе са уравнотеженим скуповима података нису прикладне за неуравнотежене скупове података. То је зато што имају снажну предрасуду према већинској класи и осетљиви су на класна искривљења [16]–[19]. На пример, мера тачности (accuracy) није прикладна за проблем неуравнотежених скупова података [20]. Ако се узме у обзир скуп података који садржи само 1% случајева мањина и 99% случајева већине, тачност је 99% ако су сви већински примери добро класификовани. Међутим, погрешно класификовани случајеви мањина од 1% могу довести до огромних трошкова, а тачност од 99% може бити катастрофа за медицинску дијагнозу. Сходно томе, друге мерне вредности су неопходне за мерење перформанси класификатора. Неке мере се издвајају директно из матрице конфузије. Они независно мере перформансе класификације већинске и мањинске класе. Неки други се комбинују за мерење перформанси класификатора. Они су описани у наставку.

1. **Прецизност (Precision)**

То је мера тачности [21]. Представља проценат добро класификованих мањинских случајева у односу на све инстанце чија је предвиђена класа мањина. Дефинисано је у једначини:

$$\text{Precision} = \frac{TP}{TP + FP} \qquad 6.2$$

2. **Опозив (Recall)**

То је проценат мањинских случајева који су добро класификовани као припадници мањинске класе. У литератури овај показатељ има неколико назива попут осетљивости, истинске позитивне стопе (TPrate) или позитивне тачности[22]. Дефинисано је у једначини:

$$\text{Recall} = \frac{TP}{TP + FN} \qquad 6.3$$

3. **Специфинчност (Specificity)**

Проценат већинских случајева који су добро класификовани као припадници већинске класе. Ова мера је такође позната као истинска негативна стопа (TNrate) или негативна тачност. Дефинисано је у једначини :

$$\text{Specificity} = \frac{TN}{TN + FP} \qquad 6.4$$

4. **Тачно позитивна стопа (False-positive rate (FPrate))**

То је проценат већине случајева који су погрешно класификовани као припадници мањинске класе. Дефинисано је у једначини:

$$\text{FPrate} = \frac{FP}{FP + TN} \qquad 6.5$$

5. **Тачно негативна стопа (False-negative rate (FNrate))**

То је проценат мањинских случајева који су погрешно класификовани као припадници већинске класе. Дефинисано је у једначини:

$$\text{FNrate} = \frac{FN}{FN + TP} \qquad 6.6$$

6. **Г-средње (G-mean)**

Указује на равнотежу између класификационих перформанси на већинским и мањинским класама [22]. Лоше перформансе у предвиђању позитивних случајева довешће до ниске вредности Г-средње вредности чак и ако негативне инстанце модел правилно класификује [23]. Неколико истраживача користило га је за процену класификатора на неуравнотеженим скуповима података [23]–[25].Г-Меан истовремено узима у обзир опозив и специфичност. Дефинисано је у једначини. Ова метрика ће се користити за тестирање приступа:

$$\text{G} - \text{Mean} = \sqrt{\text{Recall} * \text{Specificity}} \qquad 6.7$$

7. Ф – мера (F-measure)

Дефинисано је као хармонијско средство прецизности и опозива [26]. Његова вредност се пропорционално повећава са повећањем прецизности и опозива; висока вредност Ф-мере указује на то да модел боље делује на мањинској класи. Ова метрика је дефинисана у једначини:

$$\mathrm{F-Measure} = \frac{2 * \mathrm{Recall} * \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}} \qquad 6.8$$

8. ROC крива - (Receiver operating characteristic curve (ROC))

ROC крива [26], [27] је техника за визуелизацију, организацију и избор класификатора на основу њихових перформанси. Дуго се користи у откривању сигнала да представља компромис између стопе успеха и стопе лажних аларма класификатора. То је дводимензионални граф где се TPrate исцртава на у-оси, а FPrate на х-оси. За дискретни класификатор производи се пар (FPrate, TPrate) који одговара једној тачки у простору ROC. Међутим, пробабилистички класификатор даје непрекидну нумеричку вредност. Према томе, праг се може користити за стварање низа тачака у простору ROC како би се произвела крива уместо једне тачке.

9. Површина испод РОК криве (Area under the ROC curve (AUC))

Из ROC криве се дефинише још једна мера која се назива површина испод криве (AUC) [27]дефинисана у једначини 6.9 за упоређивање перформанси два класификатора. Ако је површина повезана са класификатором C1 већа од оне повезане са класификатором C2, тада су перформансе C1 боље од C2:

$$AUC = \frac{\mathrm{TPrate} + \mathrm{TNrate}}{2} = \frac{1 + \mathrm{TPrate} - \mathrm{FPrate}}{2} \qquad 6.9$$

7. Методе обраде на нивоу података

Неуједначени скупови података су у томе што су стандардни алгоритми учења класификације често пристрасни према већинским класама (познатим као „негативни“) и стога постоји већа стопа погрешне класификације у случајевима мањинских класа (названим „позитивна“ класа)[28], [29]. Због тога су током последњих година предложена многа решења за решавање овог проблема, како за стандардне алгоритме учења, тако и за ансамбл технике [30], [31].Они се могу сврстати у три главне групе [32]–[34]:

Узорковање података: у којем су примерци обуке модификовани на такав начин да дају уравнотеженију дистрибуцију класа која класификаторима омогућава да раде на сличан начин као и класификација.

Алгоритамска модификација: овај поступак је усмерен на прилагођавање основних метода учења како би се више прилагодили питањима неравнотеже

Трошковно осетљиво учење: ова врста решења укључује приступе на нивоу података, на алгоритамском нивоу или на оба нивоа заједно, узимајући у обзир веће трошкове погрешне класификације примера позитивне класе у односу на негативну класу, и стога, покушај како би се смањиле веће грешке у трошковима.

У специјализованој литератури се могу наћи поједини радове о техникама поновног узорковања које проучавају ефекат промене дистрибуције класа ради бављења неуравнотеженим скуповима података. Та су дела емпиријски доказала да је примена корака предобраде ради уравнотежења дистрибуције класа обично корисно решење [35]–[37]. Поред тога, главна предност ових техника је у томе што су независне од основног класификатора.

Технике поновног узорковања могу се сврстати у три групе или породице [38]:

• **Методе подузорковања**, које стварају подскуп изворног скупа података уклањањем инстанци (обично инстанце већинске класе).

• **Методе прекомерног узорковања**, које стварају надсет оригиналног скупа података реплицирањем неких инстанци или стварањем нових инстанци од постојећих.

• **Хибридне методе**, које комбинују оба приступа узорковања.

Међу овим категоријама постоји неколико приедлога, гдје су најједноставније предобраде нехеуристичке методе као што су случајно подузорковање и случајно преузорковање. У првом случају, главни недостатак је тај што може одбацити потенцијално корисне податке, што би могло бити важно за процес индукције. За случајно прекомерно узорковање, неколико аутора се слаже да овај метод може повећати вероватноћу да дође до прекомерног учења, јер прави тачне копије постојећих инстанци.

7.1 Основе подузорковања и прекомерног узорковања

Као класичне методе разликује се укупно девет техника. Две од ових метода, случајно подузорковање и случајно преузорковање, нису хеуристичке методе које су у почетку биле укључене у ову евалуацију као основне методе. Међутим, оне обично постижу добре резултате у врло кратком времену одзива [38].

Случајно подузорковање (Random undersampling (RUS)): То је нехеуристичка метода која за циљ има уравнотежење дистрибуције класа насумичним уклањањем примера већинске класе да би се добио уравнотежени скуп инстанци. Коначни однос балансирања се може прилагодити.

Случајно преузорковање (Random Oversampling (ROS)): То је још једна нехеуристичка метода која има за циљ уравнотежење дистрибуције класа насумичним умножавањем примера класа мањина.

Главни недостатак случајног подузорковања је тај што се овом методом могу одбацити потенцијално корисни подаци који би могли бити важни за процес индукције. Уклањање података је критична одлука коју треба донети, па стога многи предлози за подузорковање користе хеуристику како би се превазишла ограничења нехеуристичких одлука. С друге

стране, случајно преузорковање може повећати вероватноћу да дође до прекомерног учења, јер прави тачне копије примера класе мањина. На тај начин, симболички класификатор, на пример, може да конструише правила која су наизглед тачна, али заправо покривају један поновљени пример.

Што се тиче коришћене нотације, претпоставиће се да постоји сет за обуку $T\,R$ са N инстанци који се састоји од парова $(x_i, y_i), i = 1, \ldots, N$, где x_i дефинише улазни вектор атрибута, а y_i дефинише одговарајућу ознака класе. Свака од N инстанци има M улазних атрибута и они би требали припадати позитивној или негативној класи. Нека је $\hat{E} \subseteq E$ подскуп одабраних примера који су резултирали низом технике подузорковања.

Томекове везе ((Tomek links(TL)): TL [79] може се дефинисати на следећи начин: дата су два примера $E_i = (x_i, y_i)$и $E_j = (x_j, y_j)$ где су $x_j \neq y_j$ и $d(E_i, E_j)$ растојање између E_i и E_j. Пар (E_i, E_j) назива се Томекова веза ако не постоји пример E_l, такав да је $d(E_i, E_l) < d(E_i, E_j)$ или $d(E_j, E_l) < d(E_i, E_j)$. Томек везе се могу користити као метода подузорковања, елиминишући само примере који припадају већинској класи у свакој пронађеној Томек вези.

Сажето правило најближег суседа (Condensed Nearest Neighbor Rule(US-CNN)): Хартов CNN [39] се користи за конзистентног доследног подскупа примера. Подскуп $\hat{E} \subseteq E$ је у складу са E ако користи 1-најближег суседа [40], $\hat{E}$ правилно класификује примере у E. Алгоритам за стварање подскупа $\hat{E}$ из E као методе подузорковања је US-CNN. Прво, насумично црта један пример већинске класе и све примере из мањинске класе и ставља те примере у $\hat{E}$. После тога, користите 1-NN преко примера у $\hat{E}$ да бисте класификовали примере у E. Сваки погрешно класификован пример из E се помера у $\hat{E}$. Важно је напоменути да овај поступак не проналази ни најмањи доследни подскуп из E. Идеја која стоји иза ове примене доследног подскупа је елиминисање примера из већинске класе који су удаљени од границе одлучивања, јер би ови примери могли сматрати мање релевантним за учење.

Једнострани избор (One-Sided Selection(OSS)): OSS [41] је метода подузорковања која је резултат примене Томек веза [42], а затим примене USCNN. Томек везе користе се као метода подузорковања и уклањају неправилне и граничне примере већинске класе. Гранични примери се могу сматрати „небезбедним“, јер мала количина неправилности може довести до тога да падну на погрешну страну границе одлуке. US-CNN има за циљ уклањање примера из већинске класе који су удаљени од границе одлуке. Остали примери, тј. „Сигурни“ примери већинске класе и сви примери мањинских класа користе се за учење.

US-CNN + TL: Ова хибридизација [38] је слична OSS-у, али се метода US-CNNпримењује пре Томекових веза [42]. Како је проналажење Томекових веза рачунски захтевно, рачунски би било јефтиније да се изводи на смањеном скупу података.

Правило о чишћењу суседних података (Neighborhood Cleaning Rule (NCL)) (У некој литератури као NCR): NCL [43] користи Вилсоново измењено правило најближег суседа (ENN) [44] за уклањање примера већинске класе. ENN уклања сваки пример чија се ознака класе разликује од класе најмање две од њене три NN-а. NCL модификује ENNкако би повећао чишћење података. За двокласни проблем алгоритам се може описати на следећи начин:

За сваки пример $E_i = (x_i, y_i))$ у сету за обуку пронађена су његова три најближа суседа. Ако E_i припада већинској класи, а класификација дата са три NN-а противречи првобитној класи E_i, тада се E_i уклања. Ако E_i припада мањинској класи и његова три NN-а погрешно класификују E_i, тада се NN који припадају већинској класи уклањају.

Максимизација чистоће класе (Class purity maximization (CPM)): CPM [45] покушава да пронађе пар центара, један је инстанца мањинске класе, док је други инстанца већинске класе. Користећи ове центре, он дели све инстанце на два кластера C_1 и C_2. Ако било који од кластера има мање класне нечистоће од нечистоће родитеља (Imp), тада су кластери пронађени. Нечистоћа скупа примерака је једноставно пропорција примера мањинских класа. Затим рекурзивно раздваја сваки од ових кластера у потскластере. Дакле, он формира хијерархијско груписање. Ако се нечистоћа не може побољшати, рекурзија се зауставља.

Подузорковање на основу кластера (Clusterbased under sampling(SBC)): Предложено је у [46], [47]. Узимајући у обзир да је број узорака у класи неуравнотеженог скупа података N, у њему је број узорака који припадају већинској класи N^-, а број узорака мањинске класе N^+ SBC прво групише све узорке у скупу података у К кластере. Број узорака већинске и мањинске класе је H− и H + и. Према томе, однос броја узорака већинске класе према броју узорака мањинске класе у и-том кластеру је $N_i^-/\ N_i^+$. Ако је однос N_i^- према N_i^+ у скупу података о тренингу постављен на m: 1, број изабраних узорака већинске класе у i-том кластеру приказан је у изразу (7.1):

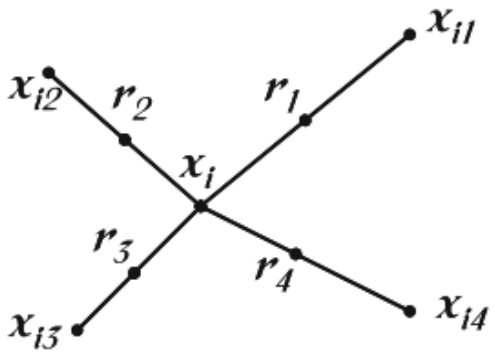
$$SN_i^- = (m \cdot N^+) \cdot \frac{N_i^-/N_i^+}{\sum_{i=1}^{K} (N_i^-/N_i^+)} \qquad 7.1$$

Приступи NearMiss: Ово је породица од четири методе засноване на информисаној хеуристици [48]. Прва метода „NearMiss-1“ бира узорке из већинске класе који су блиски неким узорцима класе мањина. На тај начин се узорци већинске класе бирају када су њихове просечне удаљености од три узорка најближе мањинске класе најмање. Друга метода „NearMiss -2“ бира узорке већинске класе, када су њихове просечне удаљености од три узорка најудаљеније мањинске класе најмање. Трећа метода „NearMiss-3“ издваја дати број узорака најближе већинске класе за сваки узорак мањинске класе. Коначно, четврта метода бира узорке већинске класе чија су средња растојања до три најближа узорка мањинске класе највећа.

7.2 Техника преузорковања синтетичких мањина (Synthetic Minority Oversampling Technique (SMOTE))

SMOTE алгоритам примењује приступ прекомерном узорковању како би избалансирао оригинални сет тренинга [49]. Уместо примене једноставне репликације примера класе мањина, кључна идеја SMOTE-а је увођење синтетичких примера. Ови нови примери настају интерполацијом између неколико позитивних случајева који су заједно. Из тог разлога се каже да је поступак фокусиран на „простор карактеристика“, а не на „простор података“. Једноставан пример овог поступка прекомерног узорковања приказан је на слици 5.10. Као основа ствара се x_i позитивна инстанца за стварање нових синтетичких тачака података. На основу метрике удаљености, неколико NN-а исте класе (тачке x_i до x_{i4})

бира се из комплета за обуку. Коначно, врши се рандомизирана интерполација како би се добиле нове инстанце r_1 до r_4.



Слика 11 - Илустрација како створити синтетичке тачке у SMOTE алгоритму

Формални поступак функционише на следећи начин. Прво се поставља укупан износ преузорковања N (целобројна вредност), који је обично дефинисан да би се добила приближна дистрибуција класе 1:1. Затим се изводи итеративни поступак који се састоји од неколико корака. Прво, инстанца инстанца се случајно бира из скупа обуке. Затим се добијају његови KNN (5 подразумевано). Коначно, N од ових K инстанци је случајно изабрано за израчунавање нових инстанци интерполацијом. Да би се то урадило, узима се разлика између вектора (узорак) који се разматра и сваког суседа. Ова разлика се множи случајним бројем извученим између 0 и 1, а затим се додаје претходном вектору обележја. То доводи до избора случајне тачке дуж „сегмента линије“ између обележја. У случају номиналних атрибута, једна од две вредности се бира насумично. Читав процес је резимиран у алгоритму.

```
1: function SMOTE(T, N, k)
   Input: T; N; k                    ▷ #minority class examples, Amount of oversampling, #NNs
   Output: (N/100) * T synthetic minority class samples
   Variables: Sample[][]: array for original minority class samples;
   newindex: keeps a count of number of synthetic samples generated, initialized to 0;
   Synthetic[][]: array for synthetic samples
2:     if N < 100 then
3:         Randomize the T minority class samples
4:         T = (N/100)*T
5:         N = 100
6:     end if
7:     N = (int)N/100   ▷ The amount of SMOTE is assumed to be in integral multiples of 100.
8:     for i = 1 to T do
9:         Compute KNN for i, and save the indices in the nnarray
10:        POPULATE(N, i, nnarray)
11:    end for
12: end function
```

Слика 12 - SMOTE алгоритам, део 1

SMOTE је послужио као инспирација за готово све методе преузорковања предложене за учење неравнотеже. У [50] аутори су истражили сва питања и даљи развој на основу SMOTE као признање 15 година од првобитног предлога.

```
1: function POPULATE(N, i, nnarray)
   Input: N; i; nnarray            ▷ #instances to create, original sample index, array of NNs
   Output: N new synthetic samples in Synthetic array
2:     while N ≠ 0 do
3:         nn = random(1,k)
4:         for attr = 1 to numattrs do          ▷ numattrs = Number of attributes
5:             Compute: dif = Sample[nnarray[nn]][attr] − Sample[i][attr]
6:             Compute: gap = random(0, 1)
7:             Synthetic[newindex][attr] = Sample[i][attr] + gap · dif
8:         end for
9:         newindex + +
10:        N − −
11:    end while
12: end function
```

Слика 13 - SMOTE алгоритам, део 2

Предности

Ублажава прекомерно учење изазвано случајним прекомерним узорковањем јер се генеришу синтетички примери, а не репликација инстанци.

Нема губитка информација.

Једноставно је применити и протумачити.

Мане

Док генерише синтетичке примере, SMOTE не узима у обзир суседне примере који могу бити из других класа. То може повећати преклапање инстанци и може увести додатне инстанце лошијег квалитета.

SMOTE није врло практичан за високо димензионалне податке.

Мотивисани недостацима првобитне SMOTE идеје, предложено је много проширења да би се поправило и добило робуснији механизам преузорковања заснован на стварању синтетичких примера.

Тренутно постоји више од 90 SMOTE проширења објављених у научним часописима и конференцијама.

8. Трошковно осетљиво учење

Трошковно осетљиво учење односи се на одређени скуп алгоритама који су осетљиви на различите трошкове повезане са одређеним карактеристикама разматраних проблема. Ови трошкови могу потицати из различитих аспеката повезаних са датим стварним животним проблемом, а пружа их стручњак или се могу научити током фазе обуке за класификатор. У литератури постоје два различита погледа на класификаторе осетљиве на трошкове. С

једне стране трошкови повезани са карактеристикама и, с друге стране трошкови повезани са класама.

Трошкови повезани са карактеристикама. Овај сценарио претпоставља да је стицање одређене особине повезано са датим трошковима, који су познати и као трошкови теста [17]. Ово се може посматрати из новчане перспективе (нпр. Особина је скупља за издвајање, јер захтевају додатна средства или лабораторијски тестови), временске перспективе (нпр. За добављање дате карактеристике треба више времена, што може проузроковати уско грло у класификацином систему) или друге потешкоће (нпр. добијање особине укључује инвазивне тестове на људима или је поступак мерења непријатан, болан или тежак за извођење) [51].

Овај аспект трошковно осетљивог учења има за циљ стварање класификатора који постиже најбоље могуће предиктивне перформансе, уз истовремено коришћење карактеристика које се могу добити уз најниже могуће трошкове (или је збир трошкова испод датог прага) [52].

Ово се може видети као вишецильно учење, где се покушава да се успостави равнотежа између перформанси и цене коришћених карактеристика [53], [54]. У многим случајевима скупље карактеристике нуде већу моћ предвиђања, што доводи до проблема да ли се користи неколико јефтинијих карактеристика или неколико скупљих [55]. Ово се такође може посматрати као задатак избора карактеристика, али многи класификатори осетљиви на трошкове (нпр. Стабла одлучивања) имају уграђен поступак оптимизације трошкова [56].

Трошкови повезани са класама. Овај сценарио претпоставља да је прављење грешака на инстанцама које потичу из одређених класа повезано са вишим трошковима. То се може посматрати из монетарне перспективе (нпр. Давање кредита особи са лошом кредитном оценом потенцијално ће проузроковати веће губитке у банци од одбијања кредита особи са добром оценом) или приоритет / здравствена / етичка питања (нпр. слање болесног пацијента кући је много скупље и опасније за болницу од додељивања додатних тестова здравој особи) [57].

Овај аспект трошковно осетљивог учења има за циљ обуку класификатора на такав начин да ће се усредсредити на класе којима су додељени већи трошкови. Они се могу сматрати приоритетима и пожељно је утицати на поступак обуке тако што ће се другачије третирати. Иако је током последње деценије учење осетљиво на трошкове добило највише пажње због проблема са небалансираном расподелом класа [37], такође се често користи у уравнотеженим сценаријима, где нетачни исходи класификације могу довести до озбиљних последица.

У контексту класне неравнотеже, трошковно осетљиво учење може се посматрати као специфична врста приступа на нивоу алгоритма [58], [59]. Претпоставља асиметричне трошкове погрешне класификације између класа, дефинисаних у облику матрице трошкова. Стандардне методе машинског учења најчешће користе такозвану функцију губитка 0–1, која исправно класификованој инстанци додељује вредност 0, а погрешно класификованој вредности 1. Тада поступак обуке има за циљ минимизирање укупних трошкова, тј. минимизирање броја погрешних предвиђања. Како функција губитка 0–1 користи исти трошак повезан са погрешном класификацијом за све разматране класе, врло је подложна искривљеној дистрибуцији класа [60]. Функција губитка 0–1 у неуравнотеженим подацима може се лако минимизирати фокусирањем на већинску класу и превиђањем (или у крајњим

случајевима чак и потпуно игнорисањем) мањинске класе. Овај проблем постаје све распрострањенији са порастом односа неравнотеже.

Методе трошковног учења су MetaCost алгоритам, трошковно осетљива стабла одлучивања, Support Vector Machines, Artificial Neural Networks и друга. С обзиром на тему рада, дато је објашњење само вештачке неуронске мреже.

8.1 Трошковно осетљиве модификације (artificial neural network (ANN-a))

Најпопуларнији поступак учења за MLP неуронску мрежу је алгоритам са пропагацијом грешке уназад, који користи скуп инстанци тренинга за подешавање слободних параметара U. Неколико радова је показало да проблем неравнотеже класе генерише неједнаке доприносе средњој квадратној грешци (MSE) током фаза обуке, где највећи допринос MSE даје већинска класа[61].

С обзиром на скуп података о обуци са две класе (J = 2) величине $N = \Sigma_j^J n_j$, где је n_j број узорака у класи j, MSEза класу j може се изразити као [61]:

$$E_j(U) = \frac{1}{N}\sum_{n=1}^{n_j} (t^n - z^n)^2 \qquad 8.1$$

где је t^n жељени излаз, а z^n стварни излаз мреже за узорак n.

Тада се целокупан MSE може записати у облику Ej(U) на следећи начин:

$$E(U) = \sum_{j=1}^{J} E_j(U) = E_1(U) + E_2(U) \qquad 8.2$$

Када је $n_1 << n_2$, тада $E_1(U) << E_2(U)$ и $\|\nabla E_1(U)\| << \|\nabla E_2(U)\|$, где оператор ∇ означава градијент функције грешке. Сходно томе, $\nabla E(U) \approx \nabla E_2(U)$. Дакле, $-\nabla E(U)$ није увек најбољи правац за минимизирање MSE у оба класе [61].

Неједнак допринос MSE може се надокнадити увођењем функције трошкова (γ) како би се избегло да MLP игнорише мањинску класу [61].:

$$\begin{aligned} E(U) &= \sum_{j=1}^{J} \gamma(j) E_j = \gamma(1)E_1(U) + \gamma(2)E_2(U) \\ &= \frac{1}{N}\sum_{j=1}^{J} \gamma(j) \sum_{i=1}^{n_j}\sum_{p=1}^{J} \left(t_p^i - z_p^i\right)^2 \end{aligned} \qquad 8.3$$

Где је $\gamma(1)\|\nabla E_1(U)\| \approx \gamma(2)\|\nabla E_2(U)\|$

У раду се помињу три различите функције трошкова [61].:

CS-MLP1 - $\gamma(j) = n_{max}/n_j$ где је n_{max} број инстанци већинске класе.

CS-MLP2 - $\gamma(j) = N/n_j$

CS-MLP3 - $\gamma(j) = \|\nabla E_{max}(U)\|/\|\nabla E_j(U)\|$ где $\|\nabla E_{max}(U)\|$ одговара већинској класи.

9. Практични примери проблема учења из небаласнираног скупа података

Следећи део рада се односи на практично решавање проблема небалансираног скупа података где ће се кроз примере приказати како и на који начин се овакви проблеми решавају, коју метрику је најбоље користити приликом учења, а на крају ће се дискутовати о резултатима[62].

9.1 Вага за мерење тежине

За овај пример користиће се синтетички скуп података назван скуп података балансиране ваге, који се може преузети из (UC Irvine Machine Learning Repository (UCI)) Репозиторијума машинског учења [63].

Овај скуп података (Слика 14) је првобитно генерисан за моделирање резултата психолошких експеримената, али је користан јер је величине којом се може управљати и има неуравнотежене класе.

Начин учитавања податка је приказан у следећем делу кода:

```
import pandas as pd
import numpy  as np

df = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-
databases/balance-scale/balance-scale.data",
                 names = ['balance', 'var1', 'var2', 'var3', 'var4'])
df.head()
```

	balance	var1	var2	var3	var4
0	B	1	1	1	1
1	R	1	1	1	2
2	R	1	1	1	3
3	R	1	1	1	4
4	R	1	1	1	5

Слика 14 - Приказ података балансиране ваге

Скуп података садржи информације (Слика 15) о томе да ли је вага у равнотежи или не, на основу тежина и растојању на крацима.

- Скуп података има једну циљну променљиву, која је означена као *balance.*
- Скуп података има 4 улазне променљиве, означене са *var1, var2, var3 и var4*

Циљна променљива има 3 класе, које су означене као:

- **R** за већу тежину на десном краку ваге (var3 * var4 > var1 * var2)
- **L** за већу тежину на левом краку ваге (var3 * var4 < var1 * var2)
- **B** за балансирану вагу (var3 * var4 = var1 * var2)

Следећи део кода приказају број инстанци у свакој класи:

```
df['balance'].value_counts()
```

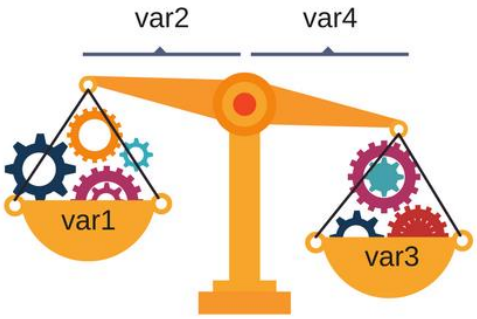
```
L    288
R    288
B     49
Name: balance, dtype: int64
```

Може се приметити да је ово пример вишекласне класификације, али извршиће се конверзија ових података тако да ово постане проблем бинарне класификације. На тај начин ће **1** представљати позитивну класу (вага је у равнотежи), док ће **0** представљати негативну класу, (вага није у равнотежи):

```
df['balance'] = [1 if b=='B' else 0 for b in df.balance]
df['balance'].value_counts()
```

```
0    576
1     49
Name: balance, dtype: int64
```

Сада се може приметити да је 8% података балансирано, према томе уколико би се увек предвидела **0**, метрика тачности (accuracy) би приказивала 92% тачно класификованих података.



Слика 15 - Визелна илустрација скупа података*[64]*

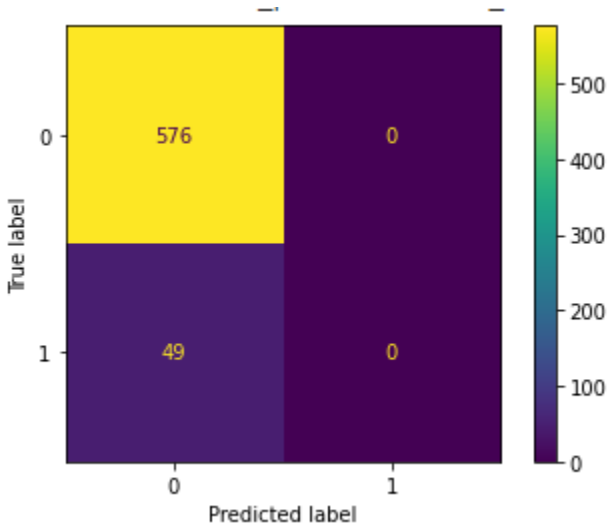
Како би се извршио тренинг користиће се Scikit-Learn библиотека[65] и неуронска мрежа која има има 4 улазна неурона у улазном слоју, 20 неурона у скривеном слоју и 1 неурон у изалзном слоју, такође се користи и опадајући градијент.

```
y = df.balance
x = df.drop('balance', axis = 1)

ann = MLPClassifier(solver='sgd', hidden_layer_sizes=(4, 20, 1), max_iter=
 300, random_state=1)

ann.fit(x,y)

pred_y_0 = ann.predict(x)

accuracy_score(pred_y_0, y)

0.9216
```

Добијена тачност је велика али већина инстанци из мањинске класе је погрешно класификована. Уколико се прикаже ROC крива (Слика 17) за дату класификацију уз помоћ следећег кода за приказ ROC криве и матрице конфузије:

Матрица конфузије се може приказати уз помоћ кода:

```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(ann, x, y, values_format = 'd')
```



Слика 16 - Матрица конфузије за небалансиране податке

У матрици конфузије са такође може приметити да је тачно предвиђено 576 инстанци за једну класу и 49 инстанци које су погрешно класификовани, док је 0 лоше класификовано за прву класу и 0 добро класификовано за другу класу.

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

Прво је потребно све предвидети вероватноћу тачно и нетачно класификованих инстанци и одабрати само оне који су вероватно тачно класификовани:

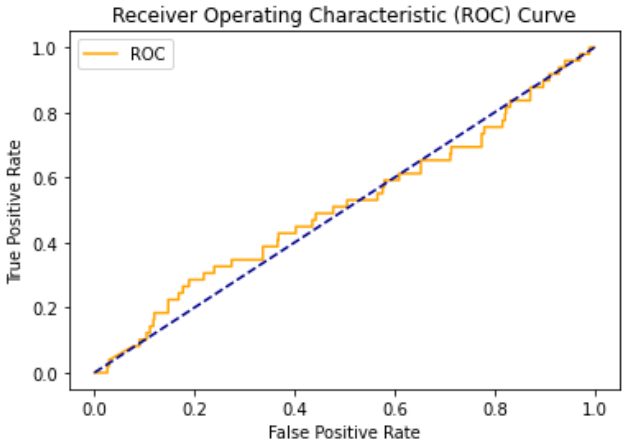
```
probs = probs[:,1]
probs = ann.predict_proba(x)
```

Потом се може приказати AUC вредност, према:

```
auc = roc_auc_score(y, probs)
print('AUC: %.2f' % auc)
AUC: 0.51
```

И на крају представити ROC криву према:

```
fpr, tpr, thresholds = roc_curve(y, probs)
plot_roc_curve(fpr, tpr)
```



Слика 17 - ROC крива дате класификације

За крај се може обратити пажња на то колико је у ствари класифкатор вредности пријавио помоћу:

```
print( np.unique( pred_y_0 ) )
array([0])
```

Овде се најбоље може видети да дати модел предвиђа само једну класу и комплетно игнорише мањинску класу. Према томе, прво предложено решење је повећати број узорака мањинске класе.

Како би се најбоље увидео проблем и квалитет тренинга, примењена је додатна метрика као што су: тачност, прецизност, одзив и ф-мера, од којих је само прва приказала вредност од 0.92 док су остале приказале 0 из разлога јер мрежа није препознала другу класу.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

# evaluate predictions
print('Accuracy: %.3f' % accuracy_score(y, pred_y_0))
print('Precision: %.3f' % precision_score(y, pred_y_0))
print('Recall: %.3f' % recall_score(y, pred_y_0))
print('F-measure: %.3f' % f1_score(y, pred_y_0))

Accuracy: 0.922
Precision: 0.000
Recall: 0.000
F-measure: 0.000
```

Повећање броја инстанци у мањинској класи:

Узорковање је поступак насумичног дуплирања запажања из мањинске класе како би се ојачао њен сигнал. Постоји неколико хеуристичих метода за то, али најчешћи начин је једноставно узорковање заменом.

Прво ће се из sklearn увести модул за поновно узорковање:

```
from sklearn.utils import resample
```

Даље ће се створити нови скуп података са узоркованом мањинском класом. Ево корака:

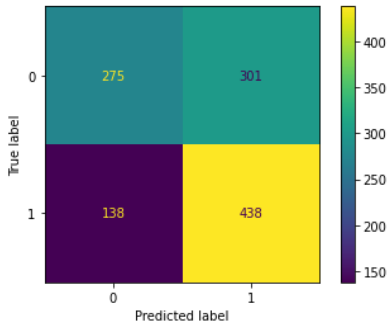
1. Прво ће се раздвојити запажања из сваке класе у различите оквире података.
2. Следеће, преузмају се узорци мањинске класе заменом, подешавајући број узорака тако да одговара већинској класи.
3. На крају се комбинује узорковану мањинска класу са оригиналном већинском класом.

```
df_majority = df[df.balance==0]
df_minority = df[df.balance==1]

df_minority_upsampled = resample(df_minority,
                                 replace=True,
                                 n_samples=576,
                                 random_state=123)

df_upsampled = pd.concat([df_majority, df_minority_upsampled])


df_upsampled.balance.value_counts()


1    576
0    576
Name: balance, dtype: int64
```

Након додавања инстанци у мањинску класу, поновним тренирањем мреже се добија резултат тачности од 0.61% што је доста приближније реалној ситуацији.

```
y = df_upsampled.balance
X = df_upsampled.drop('balance', axis=1)

ann = MLPClassifier(solver='sgd', hidden_layer_sizes=(4, 20, 1), ran-
dom_state=254)
ann.fit(X,y)
pred_y_2 = ann.predict(X)
print( np.unique( pred_y_2 ) )
print( accuracy_score(y, pred_y_2) )

[0 1]
0.6189236111111112
```

Матрица конфузије је сада:



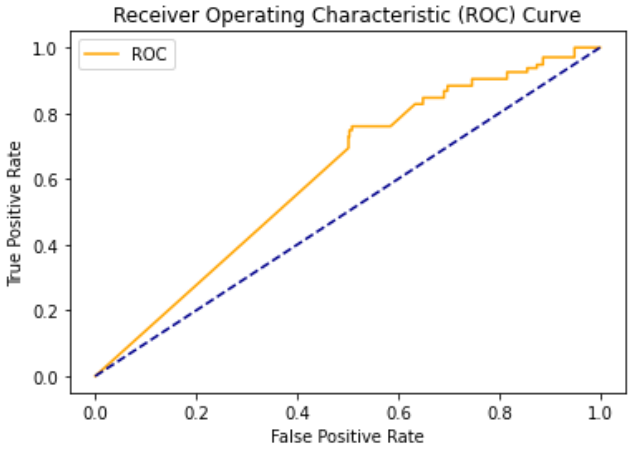
Слика 18 - Матрица конфузије приликом повећања инстанци мањинске класе

Из матрице се може видети да је 275 инстанци добро класификовано за класу 0, док је 438 добро класификовано за класу 1. 301 је лоше класификовано за класу 0 и 138 је лоше класификовано за класу 1.

Помоћу следећег кода се добија ROC крива (Слика 19):

```
probs = ann.predict_proba(X)
probs = probs[:,1]
auc = roc_auc_score(y, probs)
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y, probs)
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.61
```

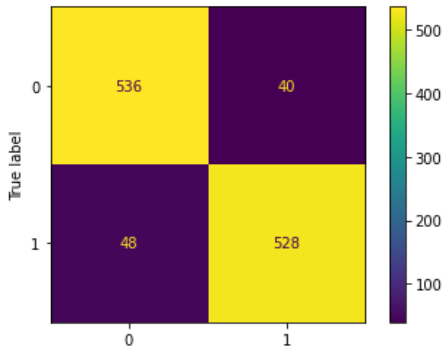


Слика 19 - ROC крива за нову мрежу

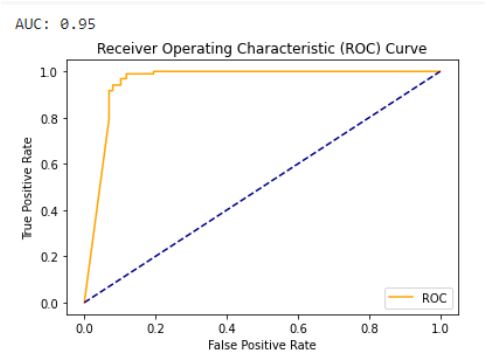
Што се тиче метрике у овом случају, мрежа је пријавила следеће:

```
Accuracy: 0.619
Precision: 0.593
Recall: 0.760
F-measure: 0.666
```

Из чега се види да квалитет процене знатно побољшан али се може знатно побољшати уколико се подеси вредност `batch_size=50` приликом тренирања мреже при чему се добија следећа метрика:



Слика 21 - Матрица конфузије приликом промене "mini_batch=50"



Слика 20 - ROC крива приликом промене "mini_batch=50"

```
Accuracy: 0.924
Precision: 0.930
Recall: 0.917
F-measure: 0.923
```

Као што се може примети тачност и други параметри су се знатно побољшали, јер променом врендноси mini_batch=50 мрежа може у мањим интервалима да учи и тиме је већа вероватноћа да наиђе на различите узорке

Смањење броја инстанци у већинској класи:

Подузорковање укључује насумично уклањање инстанци из већинске класе како би се спречило да његов сигнал доминира алгоритмом учења.

Најчешћа хеуристика за то је поновно узорковање без замене.

Процес је сличан процесу узимања узорака. Ево корака:

1. Прво ће се раздвојити запажања из сваке класе у различите скупове података.
2. Даље, извршава се поновно се узоркују интанце већинске класе без замене, подешавајући број узорака тако да одговара оном мањинске класе.
3. На крају, комбинују се узорци већинске класе са мањинском класим.

Ево програмског кода:

```
df_majority = df[df.balance==0]
df_minority = df[df.balance==1]
df_majority_downsampled = resample(df_majority,
                                   replace=False,
                                   n_samples=49,
                                   random_state=123)
df_downsampled = pd.concat([df_majority_downsampled, df_minority])
df_downsampled.balance.value_counts()


1    49
0    49
Name: balance, dtype: int64
```

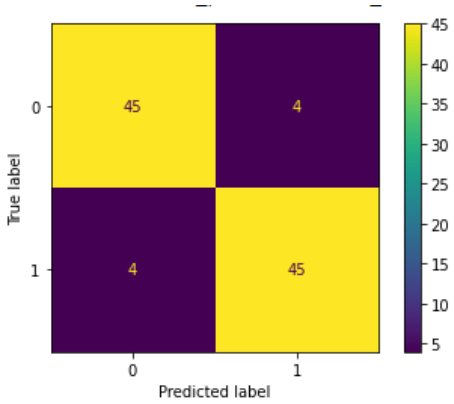
Овога пута нови скуп има мање запажања од оригинала, а однос две класе је сада 1: 1.

Поново се се врши обука помоћу неуронске мреже:

```
y = df_downsampled.balance
X = df_downsampled.drop('balance', axis=1)
ann = MLPClassifier(solver='sgd',hidden_layer_sizes=(4, 20, 1), ran-
dom_state=123, batch_size=50, max_iter=500)
ann.fit(X,y)
pred_y_2 = ann.predict(X)
print( np.unique( pred_y_2 ) )
print( accuracy_score(y, pred_y_2) )

[0 1]
0.9236111111111112
```

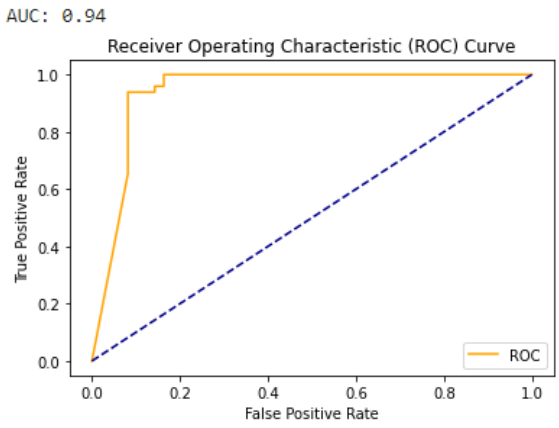
Овога пута се такође добија сличан резултат као код случаја где је повећан број инстанци мањинске класе.

Матрица конфузије је сада:



Слика 22 - Матрица конфузије приликом умањњеа инстанци из већинске класе

ROC има сличан изглед као и претходног пута:



Слика 23 - ROC крива приликом смањења броја инстанци већинске класе

При умањењу већинске класе добијени су следећи резултати:

```
Accuracy: 0.918
Precision: 0.918
Recall: 0.918
F-measure: 0.918
```

Према чему се може видети да су све 4 карактеристике мање у односу на мрежу где се користила метода увећања броја инстанци. Такође се може извести закључак да су неуронске мреже у оваквим случајевима доста непоуздане, као и да сама ROC крива не пружа значајан увид у перформансе, па је потребно прегледати и друге начине мерења перформанси мреже и поредити саме резултате.

9.2 Откривање превара са кредитним картицама

За следећи пример ће се користити сет података за детекцију лажних трансакција уз помоћ кредитних картица[66].

Важно је да компаније са кредитним картицама могу да препознају лажне трансакције кредитним картицама, тако да купцима не наплаћују ставке које нису купили.

Скупови података садрже трансакције које су европски власници картица извршили у септембру 2013. године. Овај скуп података представља трансакције које су се догодиле у два дана, где се примећују 492 преваре од 284.807 трансакција. Скуп података је изузетно неуравнотежен, позитивна класа (преваре) чини 0,172% свих трансакција.

Садржи само нумеричке улазне променљиве које су резултат PCA (Principal Component Analysis) трансформације. Нажалост, због проблема са поверљивошћу није могуће прегледати оригиналне карактеристике и додатне информације о подацима. Карактеристике V1, V2, ... V28 су главне компоненте добијене помоћу PCA, једине карактеристике које нису трансформисане помоћу PCA су „Време“ и „Количина“. Функција „Време“ садржи секунде протекле између сваке трансакције и прве трансакције у скупу података. Колона „Износ“ је износ трансакције. „Класа“ је променљива која одговора вредности 1 у случају преваре и 0 у супротном[66].

Слика *24* приказује дати сет података:

4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777

284807 rows × 31 columns

Слика 24 - Подаци о преварама платним картицама

Следећи део кода приказају број инстанци у свакој класи:

```
df['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

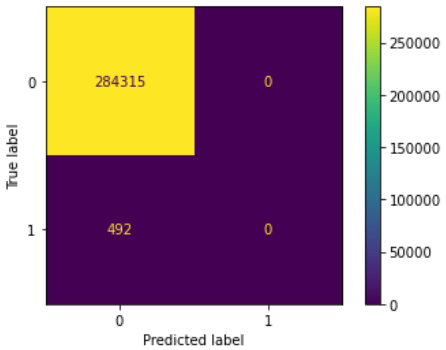
Овде се може приметити да је само 0.172% података позитивно.

Уколико се сада ови подаци ставе на улазе у неронску мрежу, без икакве претходне обраде података или надомештања истих, према следећем:

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

y = df.Class
x = df.drop('Class', axis = 1)
ann = MLPClassifier(solver='sgd', hidden_layer_sizes=(30, 20, 1),verbose=1)
ann.fit(x,y)
pred_y_0 = ann.predict(x)
```

Може се приметити да је тачност велика, чак 0,998% што је одлично. Али пошто се зна да се ради о подацима који су изразито небалансирани, та тачност говори само о подацима из већег скупа. То се може проверити приказивањем матрице конфузије према следећем:

```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(ann, x, y, values_format = 'd')
```



Слика 25 - Матрица конфузије за сет платних картица

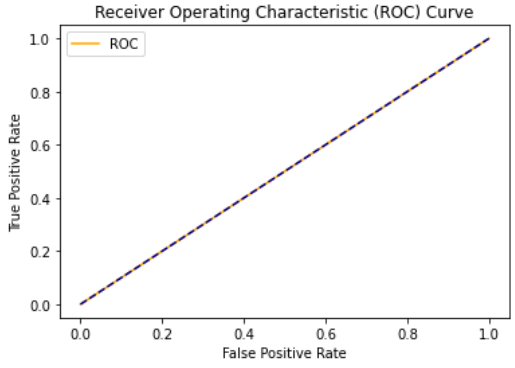
У матрици конфузије са такође може приметити да је тачно предвиђено 284315 инстанци за једну класу и 492 инстанци које су погрешно класификовани, док је 0 лоше класификовано за прву класу и 0 добро класификовано за другу класу. Овде се може видетеи да модел предвиђа само једну класу.

Штампањем ROC криве и површине испод ROC криве:

```
from sklearn.metrics import roc_curve

from sklearn.metrics import roc_auc_score

import matplotlib.pyplot as plt

def plot_roc_curve(fpr, tpr):

    plt.plot(fpr, tpr, color='orange', label='ROC')

    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.title('Receiver Operating Characteristic (ROC) Curve')

    plt.legend()

    plt.show()

probs = ann.predict_proba(x)

probs = probs[:,1]

auc = roc_auc_score(y, probs)

print('AUC: %.2f' % auc)

fpr, tpr, thresholds = roc_curve(y, probs)

plot_roc_curve(fpr, tpr)
```

Добија се да је површина испод ROC криве : AUC: 0.50

Слика приказује ROC криву:



Слика 26 - ROC крива за небаласнирани скуп података кредитним картицама

Ово је најгора ситуација. Када је AUC приближно 0,5, модел нема способност да разликује позитивну класу од негативне. Према томе се види се мора применити нека од метода која решава проблем небалансираног скупа података.

Пре него што што се изврши надомештање података, пожељно је одрадити поделу класа на класу за тренинг и класу за тест, како би се могао касније евалуирати модел на непознатим подацима.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(df.drop("Class",axis=1), df.loc[:,"Class"],
test_size=0.2)

print("Number transactions X_train dataset: ", X_train.shape)

print("Number transactions y_train dataset: ", y_train.shape)

print("Number transactions X_test dataset: ", X_test.shape)

print("Number transactions y_test dataset: ", y_test.shape)
```

Па су сада подаци:

```
Number transactions X_train dataset:  (227845, 30)

Number transactions y_train dataset:  (227845,)

Number transactions X_test dataset:  (56962, 30)

Number transactions y_test dataset:  (56962,)
```

Користиће се метода прекомерног узорковања. Најједноставнији начин прекомерног узорковања је дуплирање података у мањинској класи, али нове информације неће бити додате моделу. Алтернативно, се подаци могу синтетизовати из постојећих, који се називају прекомерно узорковање синтетичких мањина или скраћено SMOTE.

SMOTE, функционише тако што бира податке који су блиски или слични у простору карактеристика и повлачи линију између података, а нове податке прави у тачки на линији.

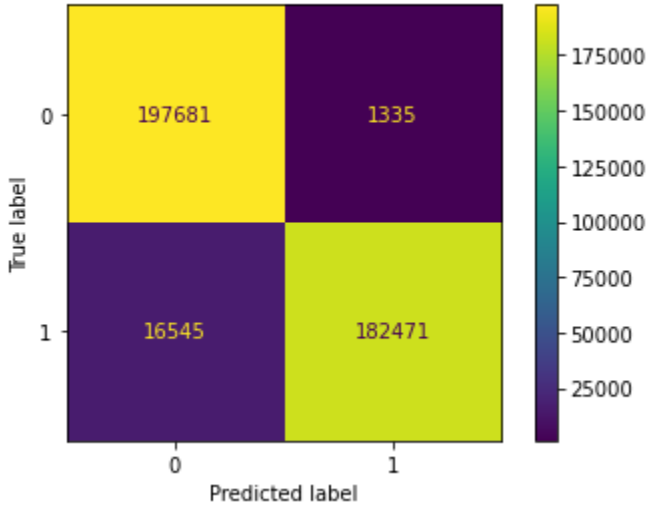
```
from imblearn.over_sampling import SMOTE

import numpy

sm = SMOTE(random_state=2)

X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())
```

Уколико се ови подаци сада доведу на улаз вишеслојне неуроснске мреже према следећем:

```
y = df.Class
x = df.drop('Class', axis = 1)
ann = MLPClassifier(solver='sgd', hidden_layer_sizes=(30, 20, 1),verbose=1)
ann.fit(X_train_res,y_train_res)
pred_y_0 = ann.predict(X_train_res)
accuracy_score(pred_y_0, y_train_res)
```

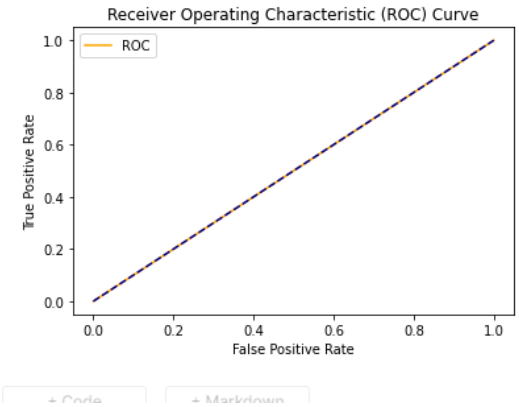
Добија се тачност од: 0,5%

Уколико се провери статистика приказивањем матрице конфузије добија се:



Слика 27 - Матрица конфузије са преузоковане податке

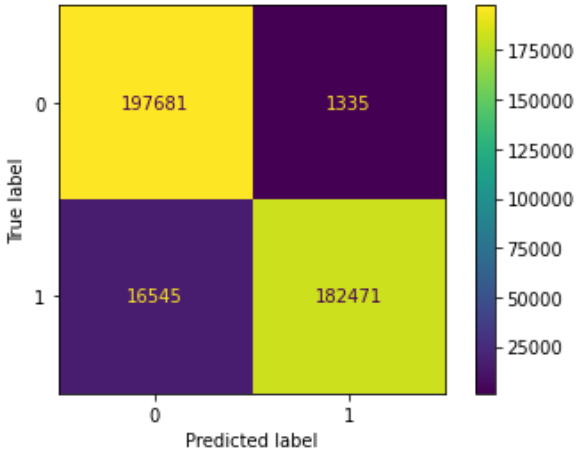
Такође се може проверити статистика и приказивањем ROC криве:



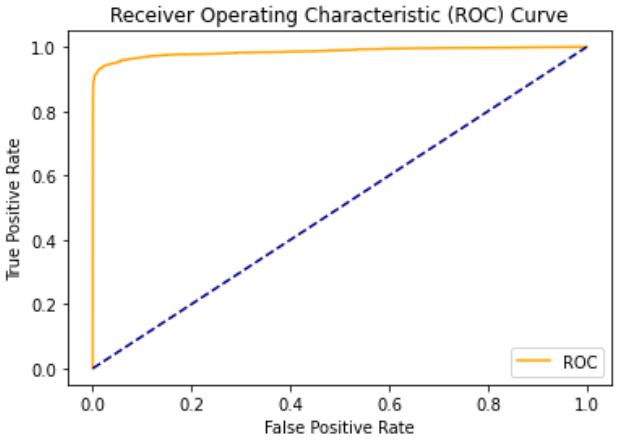
Слика 28 - ROC крива за преузорковане податке

Сада се може донети закључак да и након преузорквања SMOTE методом није се дошло до доброг резултата, могуће да је потребно другачије дефинисати неуронску мрежу пошто су подаци квалитетно надомештени. Примећено је да уносом података у други класификатор се добијају знатно бољи резултати.

Према горе наведеном, исти подаци су унети у класификатор под називом „RandomForestClassifier“ према чему се тачнос са 50% повећала на 95%. Уколико се погледа матрица конфузије и ROC крива:



Слика 29 – Матрица конфузије за "Random Forest Classifier"



Слика 30 - ROC крива за "Random Forest Classifier"

Приказом горе наведене статистике се може закључити да за овако велику разлику између класа је боље користити друге методе класификације у односу на неуронску мрежу. Разлог је тај да су одређене класификације много боље у разврставању небалансираних података.

9.3 Скуп података сонарних сигнала

Датотека садржи 111 образаца добијених одбијањем сонарних сигнала од металног цилиндра под различитим угловима и под различитим условима и 97 узорака добијених из стена под сличним условима. Пренети сонарни сигнал је фреквенцијски модулисан звучни сигнал са растућом фреквенцијом. Скуп података садржи сигнале добијене из различитих углова аспекта, у распону од 90 степени за цилиндар и 180 степени за стену[67].

Сваки образац је скуп од 60 бројева у распону од 0,0 до 1,0. Сваки број представља енергију унутар одређеног фреквенцијског опсега, интегрисану у одређеном временском периоду. Отвор за интеграцију за више фреквенције јавља се касније током времена, јер се те фреквенције преносе касније током „цвркутања“[67].

Ознака повезана са сваким записом садржи слово „R“ ако је предмет стена и „M“ ако је мина (метални цилиндар). Бројеви на лабелама се повећавају редоследом аспектног угла, али не кодирају угао директно.

9.3.1 Припрема података и тренинг

Пошто су подаци увезени у радно окружење, прво се мора погледати како ти подаци изгледају (Слика 31).

Out[42]:

	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027	0.0065	0.0159	0.0072	0.0167	0.0180	0.0084	0.0090	0.0032	R
0	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084	0.0089	0.0048	0.0094	0.0191	0.0140	0.0049	0.0052	0.0044	R
1	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232	0.0166	0.0095	0.0180	0.0244	0.0316	0.0164	0.0095	0.0078	R
2	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121	0.0036	0.0150	0.0085	0.0073	0.0050	0.0044	0.0040	0.0117	R
3	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031	0.0054	0.0105	0.0110	0.0015	0.0072	0.0048	0.0107	0.0094	R
4	0.0286	0.0453	0.0277	0.0174	0.0384	0.0990	0.1201	0.1833	0.2105	0.3039	...	0.0045	0.0014	0.0038	0.0013	0.0089	0.0057	0.0027	0.0051	0.0062	R

5 rows × 61 columns

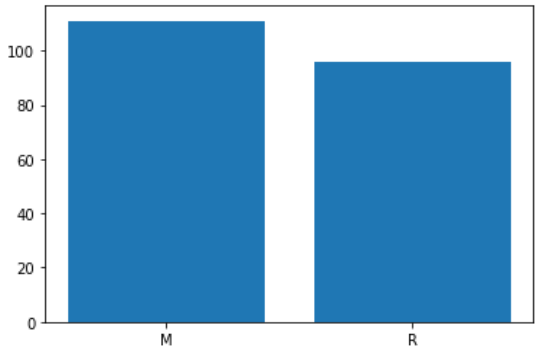
Слика 31 - Приказ података сонарних сигнала

Број инстанци које су одрежене као камен и као мина су приказане на слици испод.

```
M    111
R     96
Name: R, dtype: int64
```

Слика 32 - Број инстанци

Графички приказ броја инстанци је дат на дијаграму испод:



Слика 33 - Дијаграм број аинстанци за обе класе

Из горе приказног се може видети да су подаци уравнотежени, тј. да постоји приближно једнак број података у обе класе. Без обзира на дату ситуацију, прво ће се тестирати како класификатор учи на овим подацима, па ће се касније вештачки умањити број једне класе тако да се симулира небалансираност података. На крају ће се ти резултати поредити.

Пре тренирања ће се дати подаци поделити према следећем:

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(x,y,
test_size=0.3)


print("X_train dataset: ", X_train.shape)
print("y_train dataset: ", y_train.shape)
print("X_test dataset: ", X_test.shape)
print("y_test dataset: ", y_test.shape)
```

Тако да се добију подаци следећих величина:

```
X_train dataset:  (144, 60)
y_train dataset:  (144,)
X_test dataset:  (63, 60)
y_test dataset:  (63,)
```

Слика 34 - Сонарни подаци после поделе

Сада ће се на овим подацима извршити трениг неуронском мрежом према следећем:

```
ann = MLPClassifier(solver='adam', hidden_layer_sizes=(60, 50,
1),learning_rate = 'invscaling', max_iter= 500)

ann.fit(X_train,y_train)

pred_y_0 = ann.predict(X_test)

accuracy_score(pred_y_0, y_test)
```

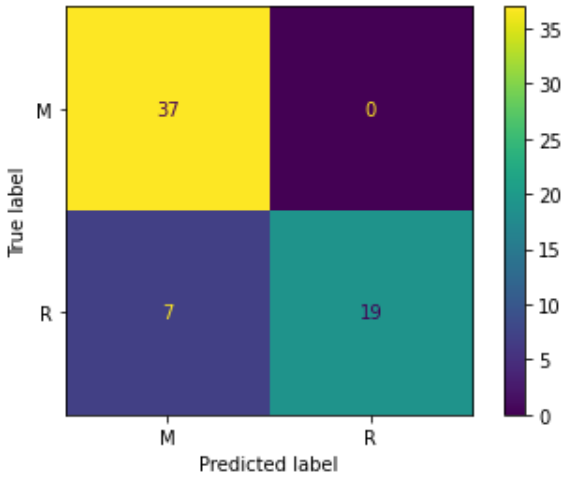
Тачност над трениираним скупом података је износила 85%, што је релативно добар резултат, који по тренутним поставкама није могуће повећати. Чак је било потребно употребити другу методу учења „Adam“, уместо до сада коришћене методе опадајућег градијента.

```
0.8571428571428571
```

Слика 35 - Тачност над тестним подацима

Уколико се прикаже матрица конфузије према следећем:

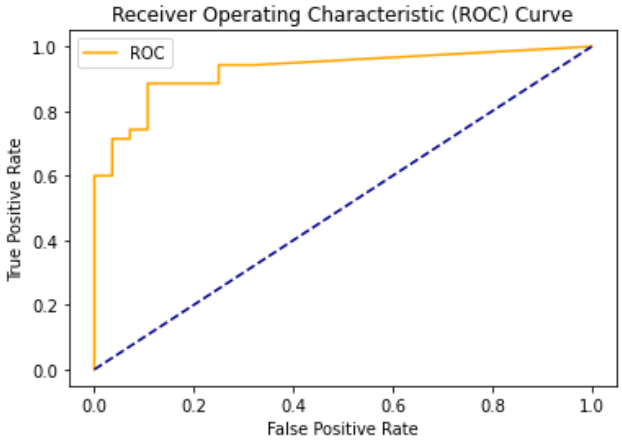
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(ann, X_test, y_test, values_format = 'd')
```



Слика 36 - Матрица конфузије за тренирани сет података

Пото ће се приказати ROC крива:

```
AUC: 0.93
```



Слика 37 - ROC крива за модел

Потребно је нагласити да је за штампање ROC криве било потрбено превести класу „R“ у bool вредности како би се крива приказала. То је изведено на следећи начин:

```
di = {"M": True, "R": False}

df = df.replace({"R": di})
```

Сада када је мрежа тренирана са пуним сетом података, може се тестирати како ће се понашати иста мрежа са једним сетом података умањеним за одређену вредност, па се аналитика касније може поредити и могуће је утврдити да ли је начин тренирања поуздан.

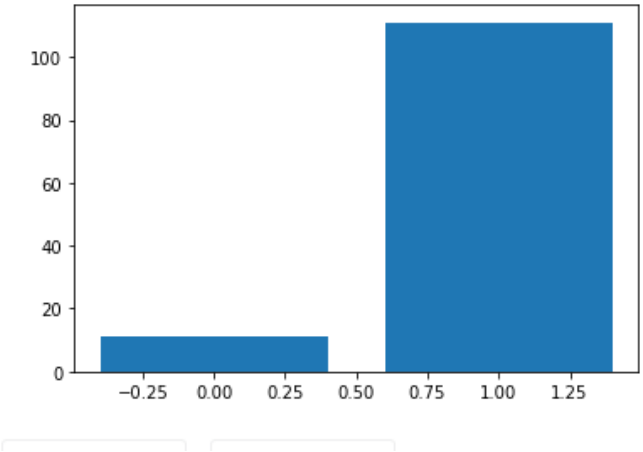
Прво је било потребно сортирати податке, тј. последњу колону како би се касније једноставније уклонио део података . Потом је узета вредносто од 85 података који ће бити уклоњени из сета података и на крају је штампана количина преосталих података у сету. То се ради према следећем:

```
df.sort_values(by=['R'])


df = df.drop(df.index[0:85])


df['R'].value_counts()
```

Нова количина података је приказана на слици испод:

```
True     111
False     11
Name: R, dtype: int64
```

Слика 38 - Преглед броја података у новом сету



Слика 39 - Граф броја података у новом сету

Први корак ка решавању датог проблема, где се у једној класи налази 111 инстанци, а у другој само 11 јесте да се оствари умањење и друге класе. Па уколико се то изврши према следећем:

```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=0)

X_resampled, y_resampled = rus.fit_resample(x, y)

y_resampled.value_counts()
```

Добиће се:

```
True     11
False    11
Name: R, dtype: int64
```

Слика 40 - Број инстанци у новој класи

Метода којом је вршено умањење једне класе је RandomUnderSampler која насумичним одабиром подскупова података за циљане класе умањује податке.

Прво ће се као и до сада извршити подела података:

```
from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test =
train_test_split(X_resampled,y_resampled, test_size=0.3)


print("X_train dataset: ", X_train.shape)

print("y_train dataset: ", y_train.shape)

print("X_test dataset: ", X_test.shape)

print("y_test dataset: ", y_test.shape)
```

```
X_train dataset:  (15, 60)
y_train dataset:  (15,)
X_test dataset:  (7, 60)
y_test dataset:  (7,)
```

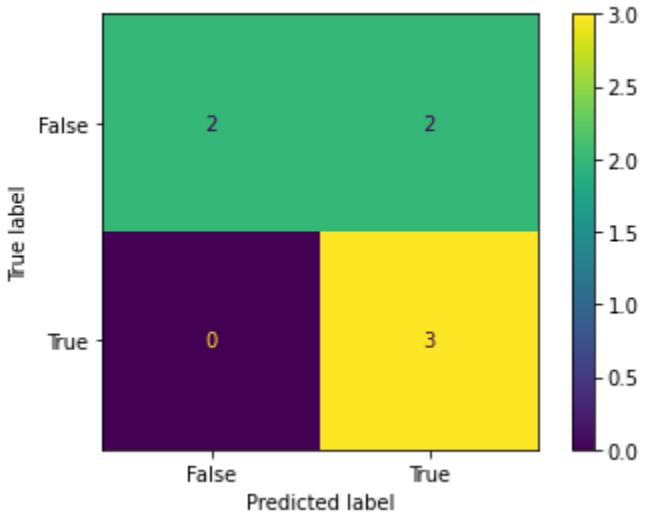
Слика 41 - Подела података на тренинг и тест

Уколико се сада изврши тренинг података над овим скупом према:

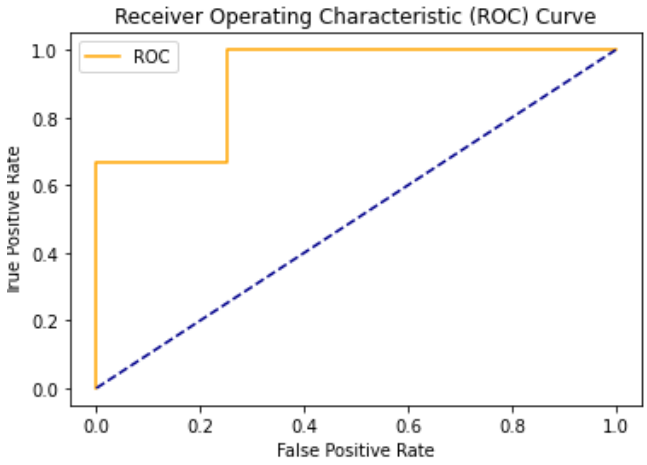
```
ann = MLPClassifier(solver='adam', hidden_layer_sizes=(60, 50,
1),learning_rate = 'invscaling', max_iter= 1000, random_state = 254)

ann.fit(X_train,y_train)

pred_y_0 = ann.predict(X_test)

accuracy_score(pred_y_0, y_test)
```

Према чему ће се добити тачност од 71%. Што је одлично за овако мали скуп података.

Такође ће се приказати и остала статистка , као што су матрица конфузије и ROC крива за дати скуп.



Слика 42 - Матрица конфузије за дати тренинг



Слика 43 - ROC крива за дати тренинг

Већ сада се да видети да треинг није био толико добар и да је вероватно дошло до превеликог прилагођавања података јер када се погледа ROC крива , види се да је она идеалног облика, што је нереално очекивати за овај тренинг.

Према томе се закључује да овакава начин тренирања где се подаци умањују није пожељан уколико се ради о врло мало скупу података јер може доћи до превелике обуке мреже.

Сада ће се одрати исти тренинг али са увећаном мањинском класом употребом ADASYN методе за преузорковање. Метода која је коришћена је увезена из библиотеке imblearn[68]**.** Па према томе позвана метода је:

```
from imblearn.over_sampling import ADASYN

X_resampled_ADASYN , y_resampled_ADASYN  = ADASYN().fit_resample(x, y)
```

Према чему се број вредности новог сета увећао и сада је:

```
True     111
False    109
Name: R, dtype: int64
```

Слика 44 - Број инстанци после примене ADASYN методе

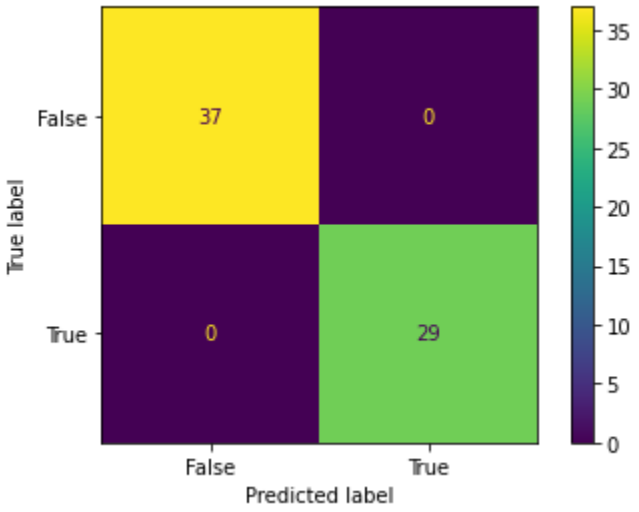
Сада ће се опет извршити подела на тест и тренинг према:

```
from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(X_resampled_ADASYN,y_resampled_ADASYN, test_size=0.3)

print("X_train dataset: ", X_train.shape)

print("y_train dataset: ", y_train.shape)

print("X_test dataset: ", X_test.shape)

print("y_test dataset: ", y_test.shape)
```

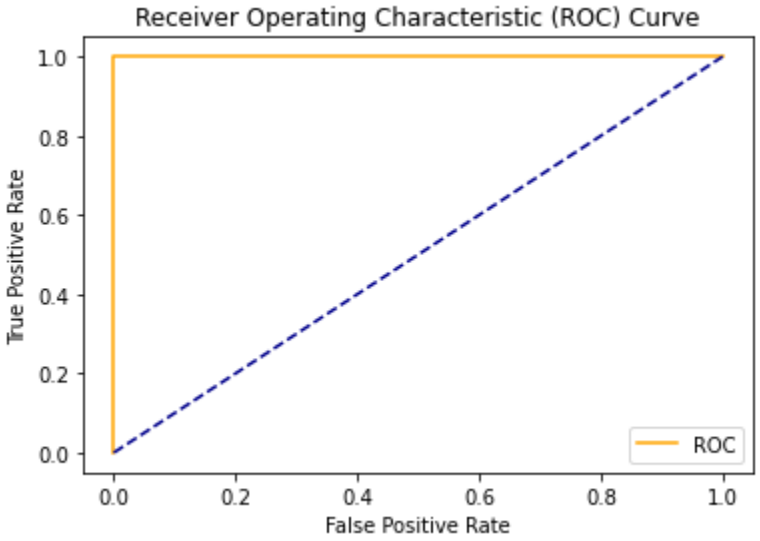
Сада ће се извршити поновни тренинг на подаци према истој медоти:

```
ann = MLPClassifier(solver='adam', hidden_layer_sizes=(60, 50,
1),learning_rate = 'invscaling', max_iter= 500, random_state = 254)

ann.fit(X_train,y_train)

pred_y_0 = ann.predict(X_test)

accuracy_score(pred_y_0, y_test)
```

При завршеном тренингу дошло се до резултата, тачније тачности од 100% која је проверена на тестним подацима, што је заправо изузетно добар резултат. Уколико се погледа матрица конфузије, види се да сви подаци тачно класификовани.



Слика 45 - Матрица конфузије над тестним подацима



Слика 46 - ROC крива над тестним подацима

10. Закључак

Као што је већ речено на почетку рада, учење са неуравнотеженим подацима односи се на сценарио у којем количине инстанци које су садржане у скупу података прате различиту расподелу, тачније број инстанци једне класе се знатно разликује од броја инстанци који се налазе у другој класи.

Рад се бавио приказом решења која могу да реше проблем неуравнотежене класификације. Први део рада обухвата теоријски део и фокусира се на математички аспект саме неуронске мреже, тек у другом делу рада се фокус ставља на примену неуронске мреже и решења која могу да допринесу бољем резултату приликом тренинга.

Први пример у раду је скуп података за балансирану вагу у којој је било потребно на основу задатих података утврдити да ли је вага балансирана или не. На овом примеру се прво показало да тренинг не укључује мањинску класу и да је је број предвиђених елемената за ту класу био нула, након тога се применила техника за увећање број инстанци мањинске класе у којој се број обе класе довео да сличну вредност при чему се тренинг знатно побољшао. Након тога је у раду било приказано да постоји и могућност умањења инстанци већинске класе како би се дошло до бољег резултата, што је на крају и био случај, јер се постигла слична тачност употребом ове методе.

Други пример је био скуп са преварама кредитним картицама, који је био комплекснији од претходног јер је имао значајну разлику броја инстанци у обе класе. На примеру је коришћена SMOT техника увећања броја инстанци класе која није донела побољшања у тренингу неуронском мрежом, осим да је тачност била око 50% . Потом је је променом типа класификатора у RandomForest добијено огромно побољшање у тренингу где је тачност износила 95% чиме се показало да тренинг неуронском мрежом у неким случајевима није пожељан јер постоје класификатори који су знанто бољи за ову намену.

Трећи пример је скуп сонарних сигнала који је балансиран скуп по природи, али је уврштен у рад како би се показало каква је разлика између тренинга балансираног скупа података и небалансираног. Прво су у тренинг неуронском мрежом укључени подаци који су балансирани и добио се резултат тачности од 85% што је одличан резултат, потом се уклонио одређен број инстанци једне класе како би се добио небалансиран скуп података (111 инстанци једне класе и 11 инстанци друге класе), потом се користила метода randomUnderSampler како би се умањила већинска класа и добио једнак број инстанци обе класе. Потом је извршен тренинг над нови скупом где су обе класе имале по 11 инстанци. Добијена тачнос је 71% што је одличан резултат за овако мали скуп података, али под претпоставком да је дошло до претретирања због малог скупа података. Потом се извршило преузорковање над почетним скупом података употребом ADASYN методе за преузорковање чиме се добио резулат тачности од 100% што такође значи да је вероватно дошло до претренирања.

Закључак аутора текста после ових експеримената јесте да постоје различити облици и начини узорковања података и тренинга над истим. Идеално би било пре тренинга сакупити што више података из света, а не користити наведене методе. Уколико то није могуће, најпожељније је корисити методу SMOTE за надомештање података уколико је огромна разлика између класа. Пак уколико се одлучи за умањење већинске класе пожељно је користити RadnomUnderSampler методу. Такође се мора обратити пажња на претренинрање и из тог разлога је пожељено корисити што више различитих начина праћења статистике.

11. Литература

[1] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, Jul. 2016, pp. 4368–4374, doi: 10.1109/IJCNN.2016.7727770.

[2] F. Chollet, *Deep learning with Python*. Shelter Island, New York: Manning Publications Co, 2018.

[3] J. Patterson and A. Gibson, *Deep learning: a practitioner's approach*, First edition. Sebastopol, CA: O'Reilly, 2017.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[5] M. A. Nielsen, "Neural Networks and Deep Learning," 2015, Accessed: Aug. 15, 2020. [Online]. Available: http://neuralnetworksanddeeplearning.com.

[6] J. D. Kelleher, *Deep Learning*. Cambridge, Massachusetts: The MIT Press, 2019.

[7] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition*. Packt Publishing, 2019.

[8] J. Krohn, G. Beyleveld, and A. Bassens, *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence*, 1st Edition. Boston, MA: Addison-Wesley Professional, 2019.

[9] M. Kuhn and K. Johnson, *Applied Predictive Modeling*, 1st ed. 2013, Corr. 2nd printing 2018 Edition. New York: Springer, 2013.

[10] H. He and Y. Ma, Eds., *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st Edition. Hoboken, New Jersey: Wiley-IEEE Press, 2013.

[11] "(PDF) Facetwise Analysis of XCS for Problems With Class Imbalances," *ResearchGate*. https://www.researchgate.net/publication/224574382_Facetwise_Analysis_of_XCS_for_Problems_With_Class_Imbalances (accessed Sep. 21, 2020).

[12] S. García and F. Herrera, "Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy," *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, Sep. 2009, doi: 10.1162/evco.2009.17.3.275.

[13] K. Ezawa, M. Singh, and S. W. Norton, "Learning Goal Oriented Bayesian Networks for Telecommunications Risk Management," in *In Proceedings of the 13th International Conference on Machine Learning*, 1996, pp. 139–147.

[14] C. G, H. M, S. H, H. S, and G. A, "Learning from imbalanced data in surveillance of nosocomial infection.," *Artif Intell Med*, vol. 37, no. 1, pp. 7–18, Oct. 2005, doi: 10.1016/j.artmed.2005.03.002.

[15] "(PDF) Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods," *ResearchGate*. https://www.researchgate.net/publication/224138101_Toward_Credible_Evaluation_of_Anomaly-Based_Intrusion-Detection_Methods (accessed Sep. 21, 2020).

[16] S. Daskalaki, I. Kopanas, and N. Avouris, "Evaluation of Classifiers for an Uneven Class Distribution Problem," *Applied Artificial Intelligence*, vol. 20, no. 5, pp. 381–417, Jun. 2006, doi: 10.1080/08839510500313653.

[17] Jin Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, Mar. 2005, doi: 10.1109/TKDE.2005.50.

[18] T. C. W. L, R. P. W. Duin, and A. P. Bradley, *Precision-recall operating characteristic (P-ROC) curves in imprecise*. .

[19] "(PDF) Analysis and Visualization of Classifier Performance: Comparison Under Imprecise Class and Cost Distributions," *ResearchGate*. https://www.researchgate.net/publication/2637219_Analysis_and_Visualization_of_Classifier_Performance_Comparison_Under_Imprecise_Class_and_Cost_Distributions (accessed Sep. 21, 2020).

[20] "On evaluating performance of classifiers for rare classes," *ResearchGate*. https://www.researchgate.net/publication/4005972_On_evaluating_performance_of_classifiers_for_rare_classes (accessed Sep. 21, 2020).

[21] M. Buckland and F. Gey, "The relationship between Recall and Precision," *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 12–19, 1994, doi: 10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L.

[22] M. Kubat and S. Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection," in *In Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 179–186.

[23] "(PDF) Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach," *ResearchGate*. https://www.researchgate.net/publication/220520052_Learning_from_imbalanced_data_sets_with_boosting_and_data_generation_The_DataBoost-IM_approach (accessed Sep. 21, 2020).

[24] R. Akbani, S. Kwek, and N. Japkowicz, "Applying Support Vector Machines to Imbalanced Datasets," in *Machine Learning: ECML 2004*, Berlin, Heidelberg, 2004, pp. 39–50, doi: 10.1007/978-3-540-30115-8_7.

[25] G. Wu and E. Y. Chang, "KBA: kernel boundary alignment considering imbalanced data distribution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, Jun. 2005, doi: 10.1109/TKDE.2005.95.

[26] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.

[27] "(PDF) ROC Graphs: Notes and Practical Considerations for Researchers," *ResearchGate*. https://www.researchgate.net/publication/284043217_ROC_Graphs_Notes_and_Practical_Considerations_for_Researchers (accessed Sep. 21, 2020).

[28] C. E. Brodley and M. A. Friedl, "Identifying Mislabeled Training Data," *jair*, vol. 11, pp. 131–167, Aug. 1999, doi: 10.1613/jair.606.

[29] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2010, pp. 875–886.

[30] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[31] "(PDF) Classification of imbalanced data: a review." https://www.researchgate.net/publication/263913891_Classification_of_imbalanced_data_a_review (accessed Oct. 08, 2020).

[32] R. Barandela, J. Sánchez, V. García, and E. Rangel, "Strategies for Learning in Class Imbalance Problems," *Pattern Recognition*, vol. 36, pp. 849–851, Mar. 2003, doi: 10.1016/S0031-3203(02)00257-1.

[33] N. V. Chawla, N. Japkowicz, and A. Kotcz, “Editorial: special issue on learning from imbalanced data sets,” *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 1–6, Jun. 2004, doi: 10.1145/1007730.1007733.

[34] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics,” *Information Sciences*, vol. 250, pp. 113–141, Nov. 2013, doi: 10.1016/j.ins.2013.07.007.

[35] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, “Automatically countering imbalance and its empirical relationship to cost,” *Data Min Knowl Disc*, vol. 17, no. 2, pp. 225–252, Oct. 2008, doi: 10.1007/s10618-008-0087-0.

[36] A. Estabrooks, T. Jo, and N. Japkowicz, *1 A Multiple Resampling Method for Learning from Imbalanced Data Sets*. .

[37] V. García, J. S. Sánchez, and R. A. Mollineda, “On the effectiveness of preprocessing methods when dealing with different levels of class imbalance,” *Knowledge-Based Systems*, vol. 25, no. 1, pp. 13–21, Feb. 2012, doi: 10.1016/j.knosys.2011.06.013.

[38] “(PDF) A Study of the Behavior of Several Methods for Balancing machine Learning Training Data,” *ResearchGate*. https://www.researchgate.net/publication/220520041_A_Study_of_the_Behavior_of_Several_Methods_for_Balancing_machine_Learning_Training_Data (accessed Sep. 21, 2020).

[39] P. Hart, “The condensed nearest neighbor rule (Corresp.),” *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, May 1968, doi: 10.1109/TIT.1968.1054155.

[40] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, 1967, doi: 10.1109/TIT.1967.1053964.

[41] M. Kubat, R. C. Holte, and S. Matwin, “Machine Learning for the Detection of Oil Spills in Satellite Radar Images,” *Machine Learning*, vol. 30, no. 2, pp. 195–215, Feb. 1998, doi: 10.1023/A:1007452223027.

[42] “Two Modifications of CNN,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976, doi: 10.1109/TSMC.1976.4309452.

[43] “(PDF) Improving Identification of Difficult Small Classes by Balancing Class Distribution,” *ResearchGate*. https://www.researchgate.net/publication/225132277_Improving_Identification_of_Difficult_Small_Classes_by_Balancing_Class_Distribution (accessed Sep. 21, 2020).

[44] D. L. Wilson, “Asymptotic Properties of Nearest Neighbor Rules Using Edited Data,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972, doi: 10.1109/TSMC.1972.4309137.

[45] Kihoon Yoon and Stephen Kwek, “An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics,” in *Fifth International Conference on Hybrid Intelligent Systems (HIS’05)*, Nov. 2005, p. 6 pp.-, doi: 10.1109/ICHIS.2005.23.

[46] S.-J. Yen and Y.-S. Lee, “Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset,” in *Lecture Notes in Control and Information Sciences*, vol. 344, 2006, pp. 731–740.

[47] S.-J. Yen and Y.-S. Lee, “Cluster-based Under-sampling Approaches for Imbalanced Data Distributions,” *Expert Systems with Applications*, vol. 36, pp. 5718–5727, Jan. 2006, doi: 10.1016/j.eswa.2008.06.108.

[48] M. Beckmann, N. F. F. Ebecken, and B. S. L. Pires de Lima, “A KNN Undersampling Approach for Data Balancing,” *JILSA*, vol. 07, no. 04, pp. 104–116, 2015, doi: 10.4236/jilsa.2015.74010.

[49] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *jair*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[50] A. Fernández, S. García, F. Herrera, and N. V. Chawla, “SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary,” *J. Artif. Int. Res.*, vol. 61, no. 1, pp. 863–905, Jan. 2018.

[51] “(PDF) A Survey on Graphical Methods for Classification Predictive Performance Evaluation | Ronaldo Prati and Carolina Monard - Academia.edu.” https://www.academia.edu/26746282/A_Survey_on_Graphical_Methods_for_Classification_Predictive_Performance_Evaluation (accessed Oct. 08, 2020).

[52] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, “SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory,” *Knowl Inf Syst*, vol. 33, no. 2, pp. 245–265, Nov. 2012, doi: 10.1007/s10115-011-0465-6.

[53] D. J. Drown, T. M. Khoshgoftaar, and N. Seliya, “Evolutionary sampling and software quality modeling of high-assurance systems,” *Trans. Sys. Man Cyber. Part A*, vol. 39, no. 5, pp. 1097–1107, Sep. 2009, doi: 10.1109/TSMCA.2009.2020804.

[54] G. Menardi and N. Torelli, “Training and assessing classification rules with imbalanced data,” *Data Min Knowl Disc*, vol. 28, no. 1, pp. 92–122, Jan. 2014, doi: 10.1007/s10618-012-0295-5.

[55] S. Chen, G. Guo, and L. Chen, *A New Over-Sampling Method Based on Cluster Ensembles*. 2010, p. 604.

[56] “Data Level Preprocessing Methods,” *springerprofessional.de*. https://www.springerprofessional.de/en/data-level-preprocessing-methods/16217938 (accessed Oct. 08, 2020).

[57] F. Fernández-Navarro, C. Hervás-Martínez, and P. Antonio Gutiérrez, “A dynamic over-sampling procedure based on sensitivity for multi-class problems,” *Pattern Recognition*, vol. 44, no. 8, pp. 1821–1833, Aug. 2011, doi: 10.1016/j.patcog.2011.02.019.

[58] B. Das, N. Krishnan, and D. Cook, “RACOG and wRACOG: Two probabilistic oversampling techniques,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, pp. 222–234, Jan. 2015, doi: 10.1109/TKDE.2014.2324567.

[59] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning,” in *Advances in Intelligent Computing*, Berlin, Heidelberg, 2005, pp. 878–887, doi: 10.1007/11538059_91.

[60] “(PDF) On the k-NN performance in challenging scenario of imbalance and overlapping.” https://www.researchgate.net/publication/230582783_On_the_k-NN_performance_in_challenging_scenario_of_imbalance_and_overlapping (accessed Oct. 08, 2020).

[61] M. Kukar and I. Kononenko, “Cost-Sensitive Learning with Neural Networks,” in *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98*, 1998, pp. 445–449.

[62] “8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset.” https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/ (accessed Feb. 07, 2021).

[63] “UCI Machine Learning Repository: Balance Scale Data Set.” http://archive.ics.uci.edu/ml/datasets/balance+scale (accessed Oct. 08, 2020).

[64] “How to Handle Imbalanced Classes in Machine Learning,” *EliteDataScience*, Jul. 05, 2017. https://elitedatascience.com/imbalanced-classes (accessed Oct. 08, 2020).

[65] “scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation.” https://scikit-learn.org/stable/ (accessed Feb. 07, 2021).

[66] “Credit Card Fraud Detection.” https://kaggle.com/mlg-ulb/creditcardfraud (accessed Feb. 07, 2021).

[67] “Mines vs Rocks.” https://kaggle.com/mattcarter865/mines-vs-rocks (accessed Feb. 07, 2021).

[68] *imblearn: Toolbox for imbalanced dataset in machine learning.* .