J48Consolidated WEKA paketea, adibide ezohikoen patroiak identifikatzeko tresna

Igor Ibarguren, Jesús M. Pérez eta Javier Muguerza

ALDAPA taldea. Informatika Fakultatea, Euskal Herriko Unibertsitatea (UPV/EHU)

http://www.aldapa.eus

txus.perez@ehu.eus

DOI: 10.1387/ekaia.14666

Jasoa: 2015-06-26 Onartua: 2015-09-08

Laburpena: Artikulu honetan WEKA ikasketa automatikorako tresnarako CTC algoritmoaren inplementazioa aurkeztuko da: J48Consolidated paketea. CTC algoritmoak lagin multzo bat sortzen du eta lagin guztietan dagoen ezagutza kontuan hartuta sailkapen zuhaitz bakarra eraikitzeko gai da, kasu berrien sailkapenaren azalpena galdu gabe. Gainera, lan honetan J48Consolidated paketeak sortutako zuhaitzen emaitzak aztertuko dira, errealitateko 36 sailkapen problematarako, laginketa mota desberdinetan oinarrituta, eta jatorrizko laginaren estaldura-maila desberdinekin. Emaitzek erakusten dute estaldura-maila altuek orokorrean sailkatzeko gaitasuna handitzen dutela eta %75eko lagin estratifikatuak erabiltzea dela problema hauetan aukerarik lehiakorrena.

Hitz-gakoak: Ikasketa automatikoa, Sailkapen gainbegiratua, Zuhaitz kontsolidatuak, CTC algoritmoa, Estalduran oinarritutako laginketa, J48Consolidated, WEKA.

Abstract: This article presents the implementation of the CTC algorithm for the WEKA machine learning tool, the J48Consolidated package. The CTC algorithm creates a set of samples and taking the knowledge of all samples into account, is able to build a single classification tree, keeping the explanation of how new examples are classified. In addition, this work analyzes the results achieved by trees built by J48Consolidated on 36 real world problems, using multiple sampling strategies and with different coverage values of the original sample. Results show that higher coverage values increase discriminating capacity and using stratified subsamples reduced to a 75% of the size give the most competitive results on these datasets.

Keywords: Machine Learning, Supervised classification, Consolidated trees, CTC algorithm, Coverage-based resampling, J48Consolidated, WEKA.

1. SARRERA

Ikasketa gainbegiratua (*ikasketa automatikoa*ren atal nagusietako bat) konputagailuko programak diren *sailkatzaileak* sortzeaz arduratzen den in-

formatika-arloa da. Sailkatzaile hauek, gaitasuna izaten dute ezagutza-arlo jakin bateko kasu edo adibide bat zein taldetako edo klasetakoa den aurreikusteko, hau da, sailkatzeko. Adibidez, medikuntzako diagnosia, iruzur-detekzioa edo objektuen (karaktere, aurpegi...) errekonozimendua izan daitezke gaur egun, oso zabalduta dauden ikasketa automatikoko problemak.

Normalean, sailkatze prozesuak bi fase nagusi izaten ditu: entrenamendu edo ikasketa fasea eta adibide berrien sailkapena bera (iragarpena ere deitua). Ikasketa fasea problemako adibide multzo batean oinarritzen da, eta bertan ezagutzen diren ezaugarri desberdinen balio desberdinekin deskribatzen da kasu bakoitza: pazientearen adina, sexua, gorputzaren tenperatura, sukarra duen ala ez..., gaixotasun baten diagnostikoan ari bagara, adibidez. Kasuak deskribatzeko erabiltzen den ezaugarri horietako bat berezia da, *klasea*, hain zuzen, eta bera izan ohi da ebatzi nahi den sailkapen problemaren gakoa. Diagnostikoaren kasuan, gaixotasun hori garatu den ala ez jasotzen duen ezaugarria edo aldagaia izan daiteke. Hitz gutxitan esanda, klasea den ezaugarriaren eta gainerako ezaugarri guztien arteko erlazioak aurkitzea izango da ikasketaren funtsa. Hori egiten duen prozesuari *ikasketako algoritmoa* deitzen zaio eta *sailkatzailea* izango da bere prozesaketaren ondorioa edo emaitza.

Ikasketako algoritmo asko daude, neurona-sare artifizialak, auzotasunean oinarritutako metodoak, euskarri bektoredun makinak..., baina zabalduenetakoen artean *erabaki- edo sailkapen-zuhaitzak* aipa genitzake. Hauek erabilerrazak suertatzen dira eta hainbat problematan oso emaitza onak lortu dituzte baina, ziur aski, beraien arrakastaren gako nagusia zera izan da: egiten duten sailkapenaren azalpena ematen duten sailkatzaile gutxitakoak izatea. Izan ere, prozesaketa hierarkiko batean oinarrituta, ikasketaren abiapuntua den adibide multzoa edo lagina, zatitan banatzen joaten da kasuak deskribatzen diren aldagai batek har ditzakeen balioen araberako galderen bitartez. Ahalik eta datu-partizio homogeneoen edo puruen lortzea dute helburu, zatitzeko erabilitako aldagaiak klasearekiko duen erlazioan oinarrituta. Adibidez, zuhaitzaren erroa (hau da, lagin osoa osatzen duena) <*adina*> aldagaiaren arabera banatu daiteke, 35 urte edo gutxiago duten pazienteak alde batetik, eta gehiago dituztenak bestetik. Gero, 2 multzo berri hauek, banan-banan aztertuko dira eta erabakiko da zein den aldagairik esanguratsuena zuhaitzaren adabegi horiek ere berriro banatzeko. Eta horrela jarraituko du zuhaitza garatzen, gelditzeko baldintzaren bat bete arte, uneko adabegiko kasu guztiak klase berekoak direla, adibidez.

80. hamarkadan hainbat algoritmo proposatu ziren sailkapen-zuhaitzak sortzeko. Hiru izan ziren arreta handiena jaso eta gaur egunerainoko eragina izan dutenak: CHAID [1, 2], CART [3] eta C4.5 [4] (egile beraren ID3 [5] aitzinekoan oinarrituta). Hauek dira haien artean dauden desberdintasun nagusiak: trata ditzaketen aldagai motak, orokortzeko gaitasuna handitzeko inausketa prozesaketa eta, nagusiki, aldagaiek klasearekin

duten erlazioa neurtzeko irizpidea, *split function* (χ^2(khi-karratua), *gini index* eta entropia-irabazia, hurrenez hurren). Hala eta guztiz ere, Ross Quinlanek diseinatutako C4.5a izan da gehien erabilitako bat: 4.914 bibliografia-aipamen, *ACM DL*-ren arabera, eta 26.512 *Google Scholarren* (lerro hauek idazten ari ginen unean). Izan ere, C4.5 algoritmoa datu-meatzaritzako lehenengo hamar —*the top 10*— algoritmoen artean identifikatuta dago [6], hainbat adituren arabera.

Sailkapen-zuhaitzen erabilpena hedatzen joan zen heinean ikusi zen entrenamendurako erabiltzen zuten laginarekiko oso sentikorrak zirela. Hau da, lagineko kasu gutxi batzuk aldatuta zuhaitza oso desberdinak sor zitezkeela eta, beraz, oso emaitza desberdinak lor zitezkeela sailkatzeko garaian. Alegia, ikusi zen oso sailkatzaile ezegonkorrak direla. Honi aurre egiteko, sailkatzaile desberdinak konbinatzeko bidea zabaltzen joan zen, hau da *sailkatzaile anitzak* sortzeko bidea, eta jarduera hau oso emankorra izan da: *Bagging* [7], *Boosting* [8], *Random forests* [9]... Horrelako kasu gehienetan, hainbat lagin desberdin sortuz lortzen dira hainbat sailkatzaile desberdin. *Bagging* izenaren kasuan, adibidez, entrenamenduko laginean oinarrituta hainbat lagin sortzen dira, *bootstrap* teknika erabiliz (jatorrizkoak duen adina kasu zoriz aukeratuta ordezkapenarekin[1]). Gero, lagin bakoitzetik sailkatzaile bat eratzen da, era independentean, eta ikasketako algoritmo bera erabilita (C4.5a Breimanen proposamenean). Azkenekoz, kasu bat sailkatzeko unean, zuhaitz bakoitzarekin sailkatzen da eta gehiengo sinplearen arabera gertatzen den klaserik bozkatuena izango da esleituko zaiona. *Bagging* lehenengo proposamenetako bat izanik ere, oraindik oso emaitza lehiakorrak lortzen dituen sailkatzaile anitza da, besteekin alderatuta [10].

Sailkatzaile anitz hauek, sailkapenaren zuzentasunaren edo egokitasunaren ikuspuntutik, askoz eraginkorragoak ziren banakakoak baino, baina beraien kalterako berriz, galdu egiten zuten egindako sailkapena azaltzeko gaitasuna. Izan ere, Domingosek esaten zuen bezala [11], «... gizaki orok banakako erabaki-zuhaitz bat erraz uler badezake ere, handiegia ez bada behintzat, horrelako 50 zuhaitzeko multzoa, sinpleak izanda ere, gainezka egiten dio pazientzia handiena duen gaitasunari ere».

Testuinguru honetan, ALDAPA ikerketa-taldeak *zuhaitz kontsolidatuak* sortzeko algoritmoa proposatu zuen, *CTC* (*Consolidated Tree Construction*) algoritmoa, hain zuzen [12, 13]. Algoritmo honek lagin multzo bat erabiltzen du eta zuhaitzak pausuz pausu sortzen diren heinean, hauen erabakietan oinarrituta, beste zuhaitz bat sortzen du, zuhaitz kontsolidatua deritzona, hain zuzen. Beraz, lagin guztietan dagoen ezagutza guztia kontuan

[1] Hau da, hainbat kasu errepikatuta egon daitezke eta beste batzuk, aldiz, ez dira agertuko.

hartuta, zuhaitz bakarra sortzen du, amaierako sailkatzaileak duen azaltze-gaitasuna mantenduz. WEKA, <*Waikato Environment for Knowledge Analysis*>, ikasketa automatikorako software libreko plataformarako [14] CTC algoritmoaren inplementazioa, *J48Consolidated*, aurkeztu zen barneko txosten batean [15].

Azken urte hauetan, CTC algoritmoa hainbat esparrutan erabili izan da eta, lanaren helburuaren arabera, laginketa mota desberdinak erabili dira; honako hauek erabili dira besteak beste: *bootstrap* laginak, lagin estratifikatuak, klaseen ordezkaritza edo klase-banaketa aldatutakoak... Ordea, lagin multzoak erabiltzen direnean, lagin motaz gain, zein lagin kopuru erabili ere erabaki behar izaten da. Hau erronka handia bihurtu da eta lan gehienetan beharrezkoa izan da ekorketa bat egitea lagin kopuru balio desberdinekin (lanen batean baita 3 laginetatik 200 laginetaraino ere) ikusteko zein baliok ematen duen portaerarik hoberena problemaren arabera.

Berriki, estrategia bat proposatu dugu eragozpen honi aurre egiteko, *estalduran oinarritutako laginketa* [16], hain zuzen. Estrategia honek, sortutako lagin multzoan dagoen jatorrizko laginaren kasuen ordezkaritza edo estaldura gutxieneko bat bermatzen duen lagin kopurua estimatzen du. Aipatutako lan honek, ordea, klase-banaketa aldatutako laginetan (zehazkiago, orekatuak diren laginetan) bakarrik jartzen du arreta, eta oso konparaketa zabalean (96 problema eta 22 algoritmo) oso emaitza esanguratsuak lortzen ditu. Bestalde, 2014. urtean argitaratutako txostenean [17], beste lagin motei dagokien estalduraren araberako adierazpena ere azaltzen da eta 36 datu-baseko multzo jakin baterako lortutako lagin kopurua zein den aztertzen da, problemaren ezaugarrien eta laginketa motaren arabera. Baina, berriz, txosten honetan ez da aztertzen zuhaitz kontsolidatuek zein emaitza lortuko lituzketen adierazitako lagin mota desberdinekin.

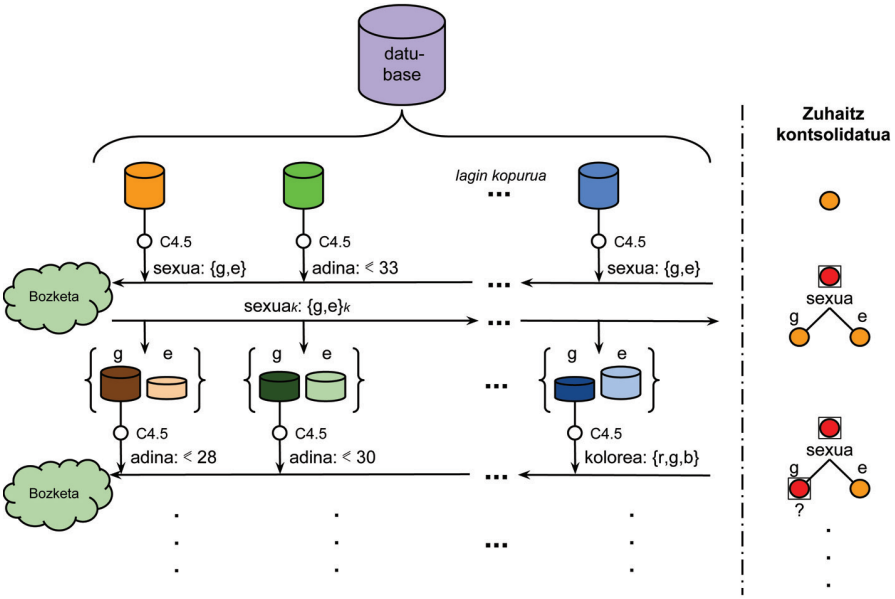
Artikulu honek helburu modura hartu du CTC algoritmorako WEKAn inplementatutako birlaginketa guztiek zein portaera duten aztertzea aipatutako [17] txostenean erabilitako 36 problemetan; hala egingo dugu lehenengo, laginketa bakoitzeko estaldura proportzio desberdinek emaitzetan duten eragina aztertuz, eta gero, laginketa desberdinek eskaintzen dituzten emaitzen diferentziei erreparatuz. Horretaz gain, lan honek argitara eramango du J48Consolidated WEKAko inplementazioarekin egindako lehenengo esperimentazio zabala.

Artikulu honen egitura ondorengo ataletan antolatuta dago. Sarrera honen ondoren, bigarren atalak J48Consolidated WEKAko inplementazioa aurkeztuko du, WEKAn hau erabiltzeko behar diren gakoei erreparatuz: inplementatutako laginketak, estalduraren gaineko adierazpenak eta J48Consolidated erabiltzeko WEKAren interfazearen nondik norakoak. Gero, J48Consolidated inplementazio honekin egindako esperimentuan arreta jartzen da, 3. atalean esperimentuaren beraren ezaugarriak deskribatuz, eta

4. atalean eskuratutako emaitzak aztertuz. Azkenik, lanaren ondorioak aurkeztuko dira, 5. atalean.

2. J48CONSOLIDATED: CTC ALGORITMOAREN WEKAKO INPLEMENTAZIOA

Sarreran esana dugu sailkatzaile anitzen antzera, CTC algoritmoak ere lagin multzo bat sortzen duela sailkatzaileak eraikitzeko, baina, azkenean, ikasketaren emaitza gisa sailkapen-zuhaitz sinple bat proposatzeko eta, beraz, gero, kasu berriak sailkatzen direnean, hauen sailkapenaren azalpena «*maneiagarria*» izan dadin. Hitz gutxitan, zuhaitz kontsolidatua eraikitzeko prozesua ondoko hau da: Zuhaitz bat joango da sortzen adabegiz adabegi lagin desberdin bakoitzarekin baina, kasu honetan, erabakitzeko zein aldagairekin banatuko den adabegi bakoitza, zuhaitz guztietako adabegi hori banatzeko zein aldagai aukeratu den kontuan hartuko da eta aldagairik bozkatuena, *aldagai kontsolidatua* esan dezagun, erabiliko da zuhaitz kontsolidatua garatzen joateko[2] (ikusi 1. irudia). Prozesu hau errepikatu egingo da zuhaitz osoan zehar, zuhaitz gehienek uneko adabegia ez banatzeko proposatu arte. Horrela, azken adabegi hau hosto bihurtuko da.



1. irudia. Zuhaitz kontsolidatu baten eraikuntzaren lehenengo urratsak.

[2] Eurek aukeratua izan ez bada ere.

Argi dago, CTC algoritmoa sailkapen-zuhaitz arruntak sortzeko algoritmo batean oinarritzen dela. Sarreran esana dugu hasiera batetik C4.5 algoritmoan [4] oinarritu genuela gure inplementazioa, algoritmoaren arrakasta eta zabalkuntza dela eta, baita ere bere inplementazioaren iturburu-kodea, C programazio-lengoaian, atzigarri zegoelako. Era berean, berriki erabakia genuen, CTC algoritmoa, algoritmo beraren detaile guztiak dokumentatzeko asmoz, software librekoa den eta oso zabalduta dagoen ikasketa automatikoko tresna batean integratzea, WEKA software-ingurunean[3] [14], hain zuzen. WEKAn jadanik integratuta zegoen C4.5 algoritmoa J48 izenarekin (hitz-joku bat eginez, WEKA Java-n inplementatuta dagoelako). Horrela, gure CTCren inplementazioa J48Consolidated klasea [15] izan zen eta, gaur egun, WEKAko pakete ofizial bat da (ikusi «*WEKA packages*»[4]).

Ondorengo azpiataletan ikusiko dugu zein laginketa mota inplementatuta dauden, hauetarako lagin kopurua nola kalkulatzen den estalduraren arabera eta J48Consolidated sailkatzailea WEKAn nola erabili, hurrenez hurren.

2.1. **Inplementatutako laginketak**

J48Consolidated klasean, hiru laginketa mota daude inplementatuta, azken urteotan zehar CTC algoritmoarekin erabili direnak. Jarraian azalduko ditugu hauek banan-bana. Gainera 2. irudian irudikatuko dira jatorrizko laginaren eta azpilagin mota bakoitzaren hainbat ezaugarri (tamainak, klasebanaketa...):

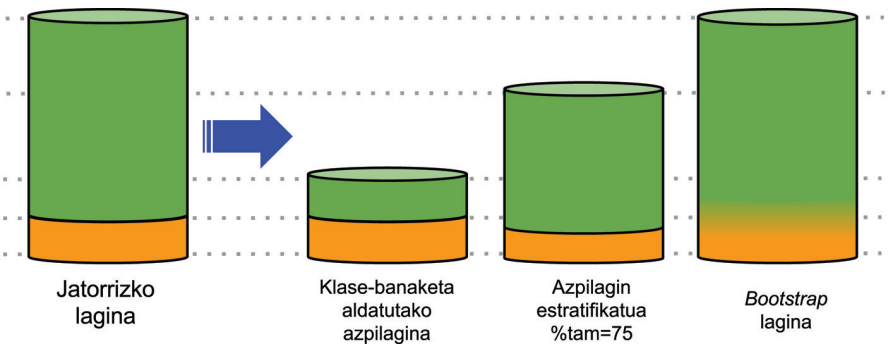
— *Klase-banaketa aldatutako laginak*. Datu-base batean zegoen klase bakoitzeko kasuen ordezkaritza aldatzea izan zen zuhaitz kontsolidatuak sortzeko arrazoi nagusietako bat. Askotan, oso nekez aurki daiteke problema batean interes handiena pizten duen klasea; oso proportzio txikian. Gaixotasun arraro baten diagnosian, adibidez, gaixotasun hori garatua duten pazienteen proportzioa %1ekoa izan liteke. Ikasketako algoritmo gehienek arazoak izaten dituzte horrelako laginekin lan egiteko eta horrek ekarri du problema mota hauek ebazteko ahalegin handiak egin behar izatea, problema mota hori izendatzeraino: *klase desorekatzearen problema* edo *<class imbalance problem>*, ingelesez.
Azpilagintzea izan da irtenbide sinple eta erabilienetako bat, hau da, gehiengoen klaseko kasu batzuk baztertzea, lagina orekatze aldera. Honek, berriz, informazio-galera handia dakar oso desorekatuak di-

[3] http://eu.wikipedia.org/wiki/Weka
[4] http://weka.sourceforge.net/packageMetaData

ren problemetan. %1eko gaixotasunaren kasuan, azpilagin orekatu batean, jatorrizko datu-basearen informazioaren, hau da *ezagutzaren* %2a baizik ez genuke lortuko. Zaila da pentsatzea informazioaren gainerako %98a ez dela baliagarria izango bi klaseen arteko erlazioak aurkitzeko. Hona hemen CTC algoritmoa diseinatzeko arrazoi nagusia: azpilaginketak dakarren informazio-galera konpentsatzea, nolabait, horrelako lagin multzoan oinarritutako sailkatzailea eraikiz.

J48Consolidated paketearen inplementazioari dagokionez, klase-banaketa aldatutako hainbat lagin sortu ahal izango dira, gutxiengoen klasearen proportzio berria adieraziz. Proportzio hau jatorrizkoarena baino handiagoa denean (adibidez, %1etik %50era iristeko) gehiengoen klasea azpilaginduko da eta, handiagoa denean, gutxiengoa bera. Kasu honetan bi klase bakarrik dituzten datu-baseei buruz ari gara. Klase anitzeko datu-basetan (2 baino gehiagokoetan), klaseak orekatzea izango da onartzen den aukera bakarra.

Mota honetako azpilaginak hainbat lanetan erabili dira zuhaitz kontsolidatuak sortzeko; kasu batzuetan, gutxiengoaren klasearen proportzio berri bat erabilita (desberdina datu-basearen arabera [13] edo beti %50a, hau da, lagin orekatuak, berriki argitaratutako lanean [16]) eta, beste batzuetan, klase-banaketa balio desberdin zabal bat erabilita [18, 19], gutxiengoaren %2-tik %98-ra, honek duen eragina emaitzetan aztertzeko.



2. irudia. Inplementatutako hiru laginketen eskema grafikoa.

Ezin utzi aipatu gabe beste laginketa metodo asko garatu izan direla, klase desorekatzearen problemari aurre egiteko eta *ekaia* aldizkarian bertan aurki daitekeela haien azterketako bat [20].

— Bootstrap *laginak*. 1. atalean esana dugu *Bootstrap* laginak direla *Bagging* sailkatzaile anitzak sortzeko erabiltzen direnak. *Bagging* erreferente garrantzitsua izan da sailkatzaile anitzetan eta saihes-

tezina zirudien CTCren portaera harenarekin alderatzea [12, 21], *Bootstrap* laginak ere erabilita.

Bootstrap delako metodoa zorizko erauzketan edo aukeraketan oinarritutako estatistika arloko teknika da. Jatorrizko laginetik zoriz joango dira kasuak ordezkapenarekin aukeratzen (hau da, erauzketa bakoitzean beti den-denak kontuan hartuko dira, behin eta berriro), jatorrizko laginaren tamainako lagin berri bat osatu arte.

Horrelako teknika erabilita, ez da kontuan hartzen inolaz ere klaseen ordezkaritza jatorrizko laginean, eta, beraz, sortutako *bootstrap* laginetan ez da mantenduko jatorrizko klase-banaketa («antzekoa» izango dela estimatu arren). Bestalde, jatorrizko laginaren kasu batzuk aukeratuak izango ez direnez, estimatzen da jatorrizkoaren kasuen %63,2 bakarrik egongo direla ordezkatuta [22].

— *Lagin estratifikatuak*. Lagin estratifikatuak ere zorizko erauzketan oinarrituta daude, baina, kasu honetan, ordezkapenik gabe. Horrek esan nahi du, jatorrizko laginetik kasu berri bat erauzten denean, kontuan hartzen direla orain arte aukeratuak izan ez direnak bakarrik. Jakina, jatorrizko laginaren tamainako beste bat eskuratu nahiko bagenu, jatorrizkoaren kopia bat lortuko genuke. Beraz, lagin desberdinak izateko, haien tamaina aldatu egiten da eta jatorrizkoaren tamainaren ehuneko (%) bat bakarrik erauzten da: %33a edo %75a, adibidez. Lagin mota hau erabiltzen duten *Bagging* moduko sailkatzaile anitzei *Subagging* deitu izan zaie.

Gainera, laginak *estratifikatuak* baldin badira (hau estatistikan erabilitako beste termino bat da), horrek esan nahi du laginketa klasez klase egiten dela, hau da, %33a erauzten dela, adibidez, klase bakoitzean eta, beraz, nahitaez, jatorrizko laginaren klase-banaketa mantentzen dela sortutakoetan.

CTC algoritmoa tankera honetako laginekin ere erabili izan da, *bootstrap* laginak erabili izan ditugun lan berberetan ([12, 21]), jatorrizkoaren %50 edota %75eko tamainak erabilita.

2.2. **Estalduran oinarritutako laginketa**

Sarreran jada esan den bezala, *estalduran* (ingelesez *coverage*an) oinarritutako laginketa berriki proposatua izan zen [16] kalkulatzeko zenbat lagin erabili behar ziren sortutako lagin multzoan estatistikoki bermatzeko jatorrizko laginaren kasuen portzentaje minimo bat nolabait jasota geratzen zela. Estimazio hau, jakina, laginketa mota bakoitzerako desberdina izango da eta lortu nahi den estaldura mailaren araberakoa.

Atal honetan, jarraian, laginketa mota desberdinetarako estalduraren adierazpena nolakoa den azaldu besterik ez dugu egingo. Hau nondik datorren zehazkiago jakin nahi izanez gero, 2014ko txostenera [17] jo daiteke.

Datozen adierazpenetan lortu nahi den estalduraren maila probabilitate balio bezala adieraziko da, hau da, balio bat 0tik 1era, kasuen %0ko estalduratik %100kora adierazteko. *Bootstrap* laginekin hasiko gara, beraiei baitagokie adierazpen bakunena.

— Bootstrap *laginak*. Kasu honetan, behar den lagin kopurua *estaldura* ren baitan bakarrik egongo da.

$$lagin\_kopurua = -\lceil \ln(1 - estaldura) \rceil \quad (1)$$

— *Lagin estratifikatuak*. Kasu honetan, estaldura mailaz gain, sortu nahi diren azpilaginen tamaina ere adierazi beharko da (*tam*), baina jatorrizko laginaren eta azpilaginen tamainen arteko erlazio bezala, 0 eta 1 artean. Hau da, adibidez 0,75 adieraziko dugu jatorrizkoaren tamainaren %75eko tamainako laginak sortu nahi baldin baditugu. Lagin estratifikatuetan klase-banaketa mantendu behar denez, lagin osoaren eta azpilaginaren tamainen arteko proportzioa eta 2 lagin hauen klase bereko tamainen arteko proportzioa proportzio bera da eta, beraz, klase baten kasuen estaldura lortzen badugu, lagin kopuru berberarekin klase guztiena ere lortuko genuke.

$$lagin\_kopurua = \lceil \log_{(1-tam)} (1 - estaldura) \rceil \quad (2)$$

— *Klase-banaketa aldatutako laginak*. Klase-banaketa aldatzen dugunean, hainbat klase edukita eta klase bakoitzaren ordezkaritzaren edozein portzentajetatik edozein portzentajetara, jatorrizko laginetik azpilaginetara aldatu nahi denean (adibidez, 3 klaseko problema batean, (%10, %20, %70) moduko laginetik orekatutako azpilaginetara (%33, %33, %33)), lagin estratifikatuen kasu orokorra dugu. Hau da, jatorrizko laginaren eta azpilaginen klase bereko tamainen arteko *d*iferentziarik *h*andieneko *k*lasean (*dhk* klasean) estaldura bermatzen badugu, gainerako klase guztietan ere hori gutxienez bermatuta geratuko da. Izan bedi tam_{dhk} jatorrizko laginaren eta azpilaginen *dhk* klaseko tamainen arteko proportzioa, 0 eta 1 artean adierazita.

$$tam_{dhk} = \frac{Azpilaginen\_dhk\_Klasearen\_Tamaina}{Laginaren\_dhk\_Klasearen\_Tamaina} \quad (3)$$

Beraz,

$$lagin\_kopurua = \lceil \log_{(1-tam_{dhk})} (1 - estaldura) \rceil \quad (4)$$

Ikuspuntu praktiko batetik, lagin kopurua kalkulatzen denean, laginketa mota edozein dela ere, kontuan hartu behar da ez duela zentzurik hiru

lagin baino gutxiago erabiltzeak zuhaitz kontsolidatu bat sortzeko. Lagin bakar batekin sortuko bagenu, C4.5 algoritmoak lagin horrekin emango lukeen zuhaitz bera itzuliko luke. Bi laginekin saiatuko bagina, adabegi bakoitzean aldagai kontsolidatua erabakitzeko garaian, edo bi zuhaitzak ados daude (eta orduan C4.5ak hartuko lukeen erabaki bera hartuko litzateke) edo, aldagai proposamen desberdinak badira, zori hutsean oinarrituko litzateke erabakia, bi aukera bakarrik kontuan hartuta. Beraz, estalduraren gaineko adierazpena erabilita, hiru baino gutxiago itzuliko balu, hiru lagin erabiltzea proposatuko litzateke.

1. taulan, atal honetan azaldutako adierazpenak erabili dira sortu beharreko lagin kopurua kalkulatzeko 4 lagin mota desberdinetarako (*Bootstrap* modukoak, estratifikatuak %75eko eta %50eko tamainekin eta lagin orekatuekin) eta estaldura 6 balio desberdin lortzeko (%-tan adierazita, %50ekotik %99,9kora). *Bootstrap* laginen eta estratifikatuen kasuan, lagin kopuruaren balioak ez daude datu-baseen ezaugarrien menpe. Estaldura baterako, edozein datu-basetarako beti eskuratuko dugu balio bera. Klase -banaketa aldatutakoetan, ordea, laginak orekatzen baditugu (gehiengoen klasea azpilaginduz), datu-basearen jatorrizko klase-banaketa ere ezaguna izan behar da. 1. taularen kasuan, lagin orekatuetarako hartutako adibidea %6,12ko gutxiengoen klasearen ordezkaritza duen bi klasetako datu-base batena da (oso datu-base desorekatua ikusteko lagin kopurua nola handitzen den estaldura maila igotzen den heinean). Gutxiengoen proportzioa esanda nahikoa da tam_{dhk} ezagutzeko; izan ere *dhk* klasea gehiengoen klasea izango da eta, nahiz eta laginen tamainarik ezagutu ez, laginaren eta azpilaginen tamainen arteko proportzioa eta hauen portzentajeen arteko proportzioa berdinak dira. Beraz, tam_{dhk} = 6,12 / (100–6,12) = %6,52.

1. taula. Estimatutako lagin kopurua estaldura balio eta laginketa mota desberdinetarako.

Laginketa mota	Estaldura					
	%50	%75	%90	%95	%99	%99,9
Bootstrap	3	3	3	3	5	7
Estratifikatua %tam = 75	3	3	3	3	4	5
Estratifikatua %tam = 50	3	3	4	5	7	10
Orekatua	11	21	35	45	69	103

1. taulan ikus daitekeen bezala, nahiko lagin kopuru txikiekin jada estaldura maila handiak lortzen dira *bootstrap* edo lagin estratifikatuekin (5 laginekin bakarrik gutxienez %99ko estaldura eskuratzen da, adibidez, *bootstrap* eta %75eko tamainako lagin estratifikatuekin). Oso desorekatua den datu-base batean, laginak orekatu nahi ditugunean, berriz, lagin kopu-

rua oso azkar igotzen da estalduraren mailarekin batera (103 lagin behar dira, kasu honetan, %99,9ko estaldura lortzeko).

2.3. **J48Consolidated-en erabilpena WEKAn**

Atal honetan azalduko dugu laburki nola erabili J48Consolidated sailkatzailea WEKA plataforman. Behin WEKAk dituen arauak jarraituz sailkapen-algoritmo bat inplementatzen dela, hura erabilgarri suertatzen da WEKAk dituen lan-ingurune edo tresna guztietan (*Explorer*-en, *Experimenter*-en...). Gehiegi ez luzatze aldera, ziur aski erabilpenik zabalduena duen tresnari, *Explorer*-i, alegia, begiratuko diogu atal honetan. *Explorer*-ek ikasketa automatikoko problema baten datu-basea kargatuta (*Preprocess* fitxa), besteak beste, sailkapen-algoritmo desberdinak (*Classify* fitxa) probatzen —*esploratzen*— lagunduko digu. Horrela, joan gaitezke sailkatzaile desberdinak aukeratzen (<*Choose*>), haien parametroak aldatzen, sailkatzailea ebaluatzeko test egokia aukeratzen, sailkatzaile bera sortzen (<*Start*>) eta, aldi berean, hark ematen dituen emaitzak aztertzen. 3. irudian sailkapen-problemak ebazteko WEKAk duen interfazea ikus daiteke, hainbat algoritmo probatu eta gero (J48Consolidated-en emaitzak dira ikus daitezkeenak, hain zuzen).

J48Consolidated sailkatzailea erabili ahal izateko, *trees* izeneko karpetan (J48rena bera) aukeratu beharko du erabiltzaileak[5]. Haren inplementazioa J48renean oinarrituta dagoenez, J48k dituen parametro guztiak heredatzen ditu (zuhaitz bat inausteko ezaugarriak, adibidez). J48ren parametroez gain, J48Consolidated-ek lagin multzoa nolakoa izan behar duen adierazteko beste bost parametro gehitzen ditu, denak <*RM*> aurrizkiz (*Resampling Method*) hasita (ikusi 4. irudia).

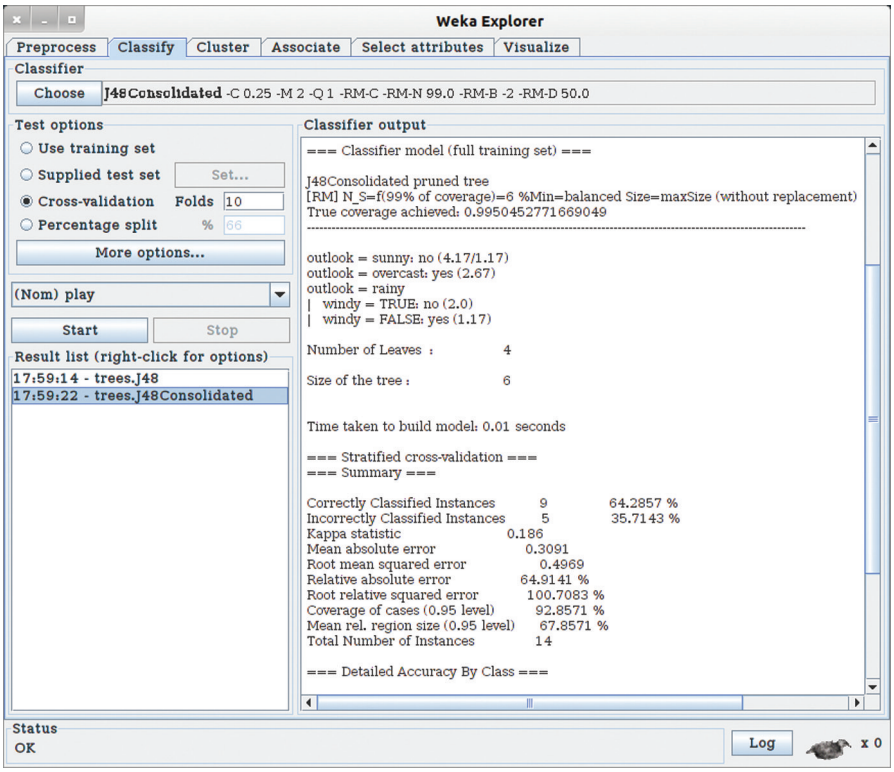
Hona hemen bost parametro hauen azalpena:

— *RMbagSizePercent*[6]. Sortu beharreko laginen tamaina adierazten du, baina entrenamenduko laginarekiko portzentaje bat bezala. Beraz, *bootstrap* laginak (%100) edo lagin estratifikatuak (nahi den portzentajean) sortzeko erabiliko dugu, *RMreplacement* parametroarekin batera jokatuz, hau da, *true* edo *false* balioekin, hurrenez hurren.
Portzentaje bat izanda, 0ren eta 100en arteko balio bat hartuko du normalean, baina beste bi salbuespenezko balio ere onartuko dira, biak klase-banaketa aldatutako laginen tamainak adierazteko oso ohizko balioak izan direnak: –1. (*sizeOfMinClass*), azpilaginek jato-

[5] Honetarako, noski, instalatuta egon beharko da. Ikusi '*downloading*' atala [17] txostenean.

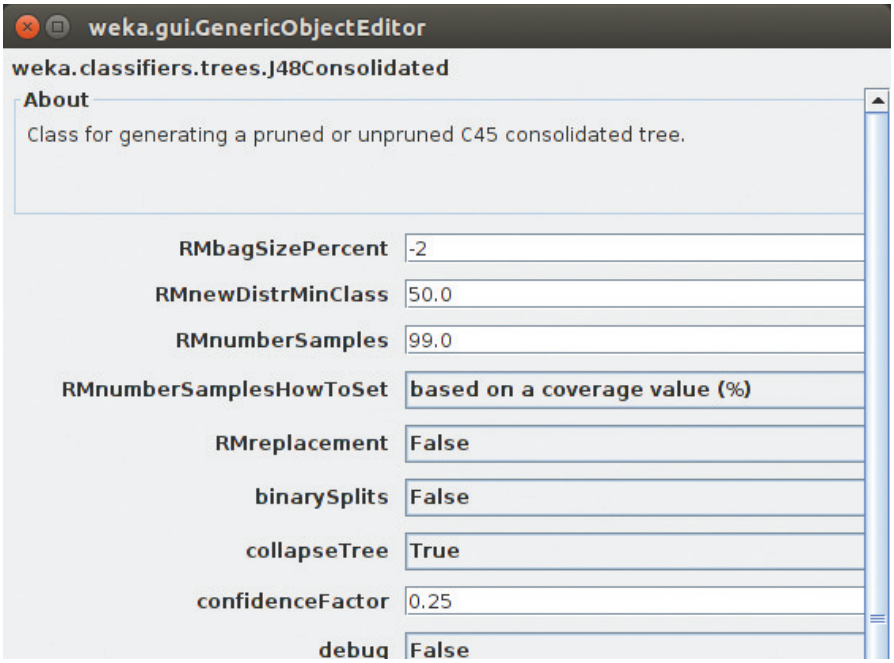
[6] Izena *Bagging*-en inplementaziotik (*meta* karpetan) hartua da, parametro hau bera baitauka, *Bagging* bera eta *Subagging* sailkatzaile anitzak eraikitzeko.

rrizko laginaren gutxiengoen klasearen tamaina eduki behar dutela adierazteko, eta –2. (*maxSize*), azpilaginek eduki dezaketen tamaina maximoa eduki dezatela (kasurik errepikatu gabe) adierazteko.



3. irudia. WEKAren Explorer lan-ingurunearen adibide bat, sailkapen-problema bat ebazten erabaki-zuhaitz desberdinekin.

— *RMnewDistrMinClass*. Hau, klase-banaketa aldatutako laginak sortu nahi ditugunean, bi klasetako problemak izanik, gutxiengoen proportzio berria adierazteko parametroa da. Beraz, 0ren eta 100en arteko balio bat izango da, normalean. Adibidez, 50 jarri beharko dugu lagin orekatuak nahi ditugunean. Klase anitzeko datu-baseetan, 50 izango da onartzen den balio bakarra, eta azpilaginak orekatuak izan daitezen adierazteko erabili beharko dena, hain zuzen.
Honetaz gain, beste bi balio ere onartuko dira: –1. (*free*), klase-banaketa kontuan ez hartzeko, eta –2. (*stratified*), jatorrizkoarena bera izateko, *bootstrap* laginak eta lagin estratifikatuak sortzeko, hurrenez hurren.



4. irudia. J48Consolidated-en parametro leihoa (partziala).

— *RMnumberSamples*[7]. Berez, lagin multzoan sortu beharreko lagin kopurua adierazten du baina bi era desberdinetan adieraz daiteke (*RMnumberSamplesHowToSet* parametroaren arabera): edo zenbaki oso positibo bat adierazita (2 baino handiagoa), edo estaldurako portzentaje bat adierazita (0ren eta 100en artean). Bigarren aukera honetan, adierazitako estaldura mailari dagokion lagin kopurua kalkulatuko da, aukeratutako laginketa motaren arabera.

— *RMnumberSamplesHowToSet*. *RMnumberSamples* parametroan lagin kopurua nola adieraziko den ezartzeko balio du: edo zuzenean lagin kopuru bat (<*using a fixed value*>), edo lortu nahi den estaldura maila (<*based on a coverage value (%)*>) adierazita.

— *RMreplacement*. Azpilaginetarako laginetik hautatu behar diren kasuak ordezkapenarekin (*true*) edo ez (*false*) hautatu behar diren adierazteko parametroa da. Ikusi dugun bezala, *true* izango da *bootstrap* laginak sortzeko eta *false* estrafikatuak edo klase-banaketa aldatutakoak sortzeko.

[7] *numIterations Bagging*-en inplementazioan.

Parametro hauen guztien balio lehenetsiak orekatuak diren laginak (tamaina maximokoak eta estaldura maila %99koa) sortzeko ezarri dira, haiek izan baitira eskaini dizkiguten emaitzarik hoberenak azkeneko lanetan, [16], gehien bat klase desorekatuetako problemetan arreta jarrita.

3. **ESPERIMENTUAREN NONDIK NORAKOAK**

Behin J48Consolidated-en oinarrizko aurkezpena eginda, lan honen helburu nagusia zera da: aztertzea zein eragin daukaten inplementatutako laginketa mota desberdinek zuhaitz kontsolidatuen emaitzetan, estaldura maila handitzen dugun heinean. Zehazki aztertuko ditugun estaldura mailak, balio baxuenetik (3 lagin) aurrera, honakoak izango dira: %10, %20, %30, %40, %50, %75, %90, %95, %99 eta %99,9 (11 balio guztira).

Konparazioan erabiliko ditugun laginketa desberdinak hauexek izango dira: *Bootstrap* laginak, azpilagin estratifikatuak lau tamaina desberdinekin (%90, %75, %50 eta %25) eta lagin orekatuak bi tamainarekin (ahal den handiena eta gutxiengoen klasearen tamainakoa, *tamMax* eta *tamGutxiengoa* deituko ditugunak, hurrenez hurren). Laginketa mota hauetako gehienak beste lan batzuetan erabiliak izan arren (estratifikatuak %90eko eta %25eko tamainakoak izan ezik), inoiz ez da (dugu) guztien artean horrelako konparaziorik egin. Bestalde, esan behar da denak aplikatu daitezkeela bi klasetakoak eta klase anitzekoak diren datu-baseetan, arazorik gabe.

Esperimentu honetan erabiliko dugun datu-base multzoa WEKAko webguneko *Data-sets* atalean[8] atzigarri dagoen lehenengo datu-base multzoa da, «*A jarfile containing 37 classification problems, originally obtained from the UCI repository*»[9]. Datu-base hauek guztiak arlo desberdinetako sailkapen-problemak dira eta *UCI*ko biltegian [23] (baina WEKAk onartzen duen *ARFF* formatuan jadanik kodetuta) dauden problemen ordezkari egokitzat har daitezke, ezaugarri desberdin askotarikokoak direnez gero: sailkatzeko arloa bera (medikuntza, industria, objektu errekonozimendua...), aldagai mota desberdinak (diskretuak, jarraikiak...), bi klasetakoak eta klase anitzekoak, klase ordezkaritza ere askotarikoa, oso desorekatuetatik oso orekatuetara, eta abar. 2. taulan 36 datu-baseren ezaugarri nagusien laburpen kuantitatiboa erakusten da, ezaugarri bakoitzaren batez besteko balioarekin, baita mediana, minimoa eta maximoa ere (datu-base multzoaren aberastasuna ikus dadin). Ikus daitekeenez, muturreko problemak aurki daitezke: gutxiengoen klasearen proportzioa %0,05ekoa duen batetik 26 klase dituen problemetara, edo 57 adibide bakarrik dituena. Datu-base guztien zehaztasunak jakin nahi izanez gero, [17] txostenera jo

[8] http://www.cs.waikato.ac.nz/ml/weka/datasets.html

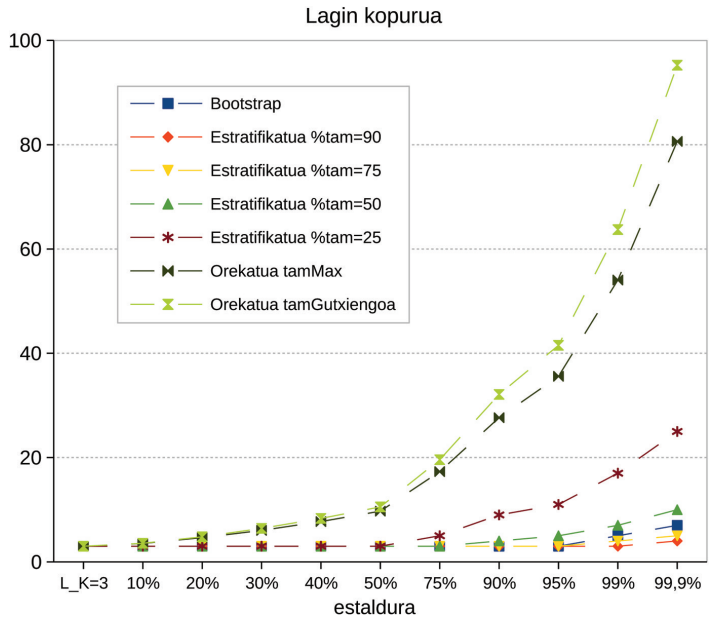
[9] Berez, 36 dira gero!

daiteke. Azkenik, azpimarratu nahi dugu artikulu honek datu-base multzo honekin eta J48Consolidated-en inplementazioarekin egindako horrelako esperimentu zabalaren lehenengo lana jasotzen duela.

2. taula. UCI biltegiko 36 datu-baseen ezaugarrien laburpena.

	#adibide	#aldagai	#klase	gutxiengoen klasea		gehiengoen klasea	
				#adibide	%	#adibide	%
Batez bestekoa	1720,53	24,64	5,47	330,22	22,79	662,64	50,90
Mediana	530,00	19,50	2,50	113,00	26,88	221,50	54,59
Minimoa	57	5	2	1	0,05	37	4,07
Maximoa	20000	70	26	3916	48,20	4208	93,88

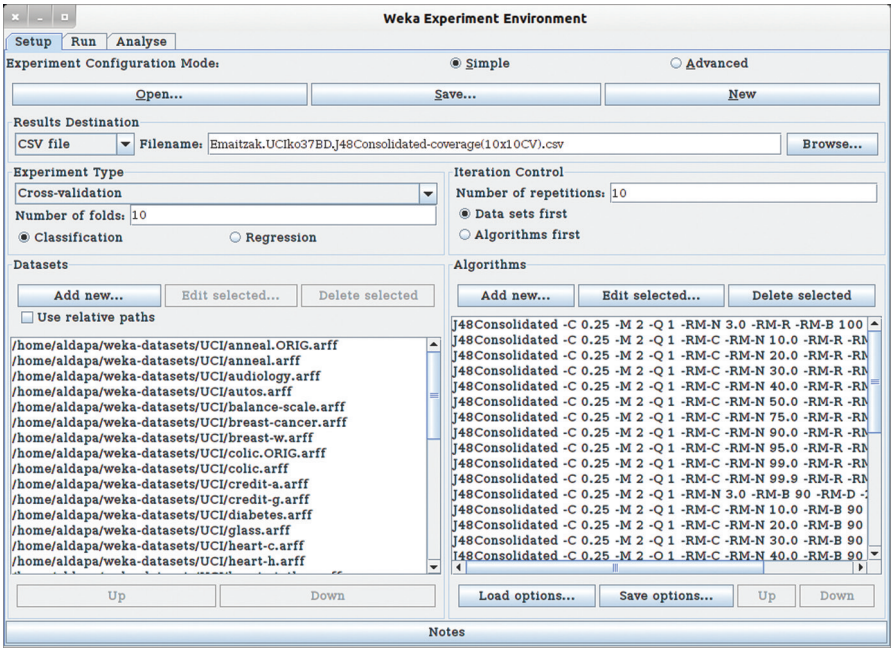
Behin datu-base multzoa aurkeztuta, 5. irudian grafikoki jasotzen da 36 datu-base hauentzat, laginketa motaren arabera, eta estaldura mailaren ekorketarako, zuhaitz kontsolidatuak sortzeko beharrezko lagin kopurua. Lagin orekatuen bi kasuetan, klaseen ordezkaritzaren araberakoak direnez, irudikatutako balioak 36 datu-baseen batez bestekoak dira. Hemen ere, gehiegi ez luzatze aldera, [17] txostenera jo daiteke xehetasunetan sartu nahi izanez gero.



5. irudia. Estimatutako lagin kopurua erabilitako estaldura balio eta laginketa motetarako.

Eraikitako sailkatzaileen portaera konparatzeko asmoz, sailkatzeko gaitasunaren ikus-puntutik, oso zabalduta dagoen *AUC* irizpidea erabiltzea erabaki dugu. AUC (*Area Under ROC Curve*) balioak sailkatzaile baten asmatze-tasaren (zehatz esanda, klase positiboaren sentikortasunaren) eta alarma faltsuen ratioaren arteko konpromisoa neurtzen du nolabait. Ingelesezko izenak adierazten duen bezala, sailkatzaile bati dagokion ROC kurbaren azaleran oinarrituta dago: seinale-detekzioaren teoriatik ikasketa automatikora ekarritako teknika bat da, hain zuzen. Hitz gutxitan esanda, teknika honek sailkatze-espazioko hainbat testuinguru desberdinetan ebaluatzen du sailkatzailearen portaera eta, beraz, oso irizpide edo neurri sendoa da sailkatzaileen eraginkortasuna neurtzeko eta hura erabiltzea gomendatu izan da beste neurri batzuen aurrean [24]. ROCen eta AUCen sakontzeko jo Fawcett-ek egindako artikulura [25].

Sailkatzaile baten portaera kuantitatiboki estimatzeko, irizpidea edozein dela ere (gure kasuan AUCa), sailkatzailea errealitatetik ahalik eta gertuen jarrita egin behar da. Errealitatean sailkatzaileak sailkatu beharko dituen kasuak, normalean ez dira sailkatzailea bera sortzeko (ikasteko) erabili diren kasu berak izango. Izan ere, ikasteko erabili dituen kasu berberak jartzen baldin badizkiogu sailkatzeko (<*Use training set*> WEKAko test aukeretan, ikusi 3. irudia), pentsatzekoa da oso emaitza onak emango di-



6. irudia. WEKAren Experimenter lan-ingurunearen leihoko adibidea egindako esperimentuaren konfigurazioarekin.

tuela; baikorregia, akaso. Baina errealitatea ez da horrelakoa izango. Beraz, estimazioa egoki egiteko, lagin osoa banatzen da entrenamendurako eta testerako; eta gainera, ez behin, baizik eta hainbat alditan. Horri *balidazio gurutzatua* deritzo (<*Cross-validation*> WEKAko test aukeretan), zeinak parametro bat onartzen duen, *Folds* izenekoa, jatorrizko lagina zenbat zatitan banatuko den adierazten duena. Horrela, 10 baldin bada (oso balio arrunta), 10 sailkatzaile desberdin sortuko dira, zati edo *fold* bat baztertu, eta beste 9 zatiak erabilita. Baztertutako zatia erabiliko da testatzeko kasu bakoitzean, eta, azkenean, 10 emaitzen batez bestekoa izango da lagin osoarekin sortuko genukeen sailkatzailearen eraginkortasunaren estimazioa. Gaur egun, gainera, hau hainbat alditan ere egiten da (zorizko partiketa desberdinekin), adibidez 5 edo 10 aldiz. Lan honetan 10 aldiz 10 *fold Cross-validation* (10×10*CV*) prozedura erabili dugu sailkatzaile guztien eraginkortasuna neurtzeko. WEKAko *Experimenter* tresnak horrelako esperimentuak egitea ahalbidetzen du, oso interfaze erabilerrazaren bidez (ikusi 6. irudian), bertan datu-base multzoa (ezkerrean) eta algoritmo multzoa (eskuinean) ezarrita.

Beraz, orain arte esandakoaren arabera, 277.200 J48Consolidated sailkatzaile (36 datu-base × 10×10CV × 7 laginketa mota × 11 estaldura maila) sortuko dira eta ebaluatuko dira lan honetan. Honetaz gain, beti bere gertueneko erreferentea izan den J48 sailkatzailea ere gehituko dugu konparaketan. Kasu honetan, laginketarik erabiltzen ez denez, sailkatzaile bat eskuratuko dugu entrenamenduko lagineko (360 guztira, 36 datubase × 10×10CV).

Azkenik, lan metodologiaren barruan aipatzeke geratu den atala *diferentzia estatistikoki esanguratsuen azterketa*ren ingurukoa da. Sailkapen-problema bat (edo multzo bat) ebazten saiatzen garenean oso ohikoa da sailkatzaile desberdinek dituzten onurak aztertzea. Sailkatzaile desberdinek emaitza desberdinak eskainiko dituzte, batzuk beste batzuk baino hobeagoak izango dira, baina zein puntutaraino dira «esanguratsuak» diferentzia horiek? Berriro beste estimazio bat egingo bagenu beste 10×10CV erabilita, iraungo lukete emaitzen gaineko ondorioek? Gaur egun, galdera hauei erantzuna emateko estatistikatik eratorritako teknikak erabiltzen dira, funtsean, *hipotesi-test*etan oinarritutakoak. Honela, esperimentu baten emaitzen gainean hipotesi-test bat planteatuko da eta horrek esango digu printzipioz hoberena geratu den sailkatzailearekiko diferentziak estatistikoki esanguratsuak diren ala ez, hau da, zoriari esker suertatu diren horrela ala ez. Test baten emaitza balio kuantitatibo batekin neurtzen da; nolabait, *hipotesi nulua* baztertzean oker egoteko probabilitatea, hain zuzen. Sailkapeneko testuinguruan, hipotesi nulua sailkatzaile desberdinen portaeran diferentziarik ez egotea izaten da eta, beraz, probabilitateko balioa (edo <*p-value*>) zenbat eta txikiagoa izan, orduan eta esanguratsuagotzat jotzen dira diferentziak sailkatzaileen artean.

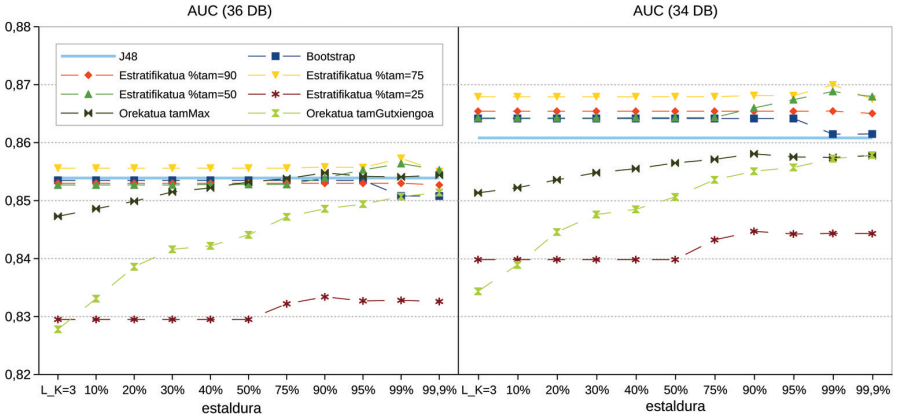
Demšar-ek 2006. urtean argitaratu zuen [26] gerora izugarrizko arrakasta izan duen diferentzia esanguratsuak aztertzeko metodoa, eta haren gainean hainbat garapen eta gomendio interesgarri proposatu izan dira beste lanetan [27]; guk jarraitu ditugunak, hain zuzen. Prozedura honen arabera, hainbat datu-baseren eta hainbat (2 baino gehiago) sailkatzaileren emaitzen gainean, lehenengo Friedman-en test ez-parametrikoa aplikatu beharra dago jakiteko ea diferentzia esanguratsurik dagoen ala ez. Test honen beste emaitza baliagarri bat zera da: alderatutako aukeren batez besteko *ranking*ak ere eskaintzen dituela, hau da, zein den hoberena edo 1.a, zein 2.a, 3.a eta horrela, azkenekoraino (okerreneraino).

Friedman-en testa aztertuta, diferentziarik baldin badago, orduan, *post-hoc* motako test bat exekutatzen da, diferentzia horiek zein sailkatzaileren artean, binaka hartuta, gertatu diren azaleratzeko. Hemen ere aukera anitz daude (bat denen kontra (1×n) edo denak denen kontra (n×n) egin nahi izatearen arabera, edo test kontserbadoreagoak edo arrisku handiagoak hartzen dituztenak aukeran...) baina, emandako gomendioei jarraiki [27], guk n×n motako Shaffer test indar handikoa aukeratu dugu lan honetarako. Bi alternatiba bakarrik konparatzen direnean, ordea, Demšar-ek Wilcoxon-en ranking zeinudunen testa (<Wilcoxon Signed-ranks test>) erabiltzea proposatzen du.

4. **EMAITZAK**

Atal honetan azaldutako esperimentuaren emaitzak erakutsiko ditugu. 7. irudian, J48-Consolidated-en aukera guztien eta J48ren batez besteko AUC balioak azter daitezke. Bertan J48Consolidated-en laginketa mota bakoitzean estaldura mailak duen eraginaren bilakaera agertzen da. J48ren emaitzak, berriz, marra zuzen batean gehitu dira (oinarrizko erreferente gisan), estaldurak eraginik ez baitu. Kurbetako puntu bakoitza 36 datu-baseen batez besteko balioa da, ezkerreko irudian, eta 34 datu-baseena eskuinekoan, «*anneal.ORIG*» eta «*colic.ORIG*» kenduta, hain zuzen. Bi datu-base hauek bi bertsio dituzte multzoan: jatorrizko bertsioa, «*.ORIG*» atzizkia dutena, eta *zuzendutako* kodeketa bati dagokiona[10]. Jatorrizko bertsioek emaitza nahasgarrietara eraman dezaketela pentsa badaiteke ere, azken finean, sailkapen-problematzat har daitezke (agian, jatorriz irudikatu nahi zituzten problemak ez irudikatuta) eta, beraz, bi datu-base multzoen azterketari ekitea erabaki dugu.

[10] Jatorrizko bertsioetan balio bakar bat hartzen zuten aldagaiak, edo aldagai jarraikiak diskretu moduan hartuta, baita aldagai errepikatuak ere, aurki daitezke.



7. irudia. AUC batez besteko balioak. 36 datu-baseenak, ezkerrean, eta 2 kenduta, eskuinean.

7. irudiko laginketa moten emaitzei erreparatuz, kasu gehienetan, estaldura maila igo ahala, AUC emaitzak hobera doazela esan daiteke. Hau oso nabarmena da orekatutako laginetan (eragina handiagoa baitauka estalduraren ideiak) eta ez hainbeste, berriz, lagin estratifikatu edo *bootstrap* laginetan (gogoratu, %tam-en arabera, oso estaldura balio handiekin bakarrik beharko dira 3 lagin baino gehiago). Arreta laginketa moten arteko konparazioan jarriz gero, batez bestean laginketa mota gehienek J48ren emaitzetara gerturatzeko joera eduki arren, lagin estratifikatuak (%50eko tamainatik gorakoak) edo *bootstrap* laginak erabiltzen dituzten zuhaitz kontsolidatuek emaitza hobeagoak lortzen dituzte (estaldura maila guztietarako eskuineko grafikoan) J48k baino. Lagin orekatuak erabiltzen dituzten J48Consolidated-ak, berriz, J48ren azpitik geratzen dira.

Ikusten da emaitza desberdinak lortzen direla aukera desberdinekin, baina zein puntutaraino dira diferentziak estatistikoki esanguratsuak? Aukera asko daudenez gero, azterketa hau era hierarkikoan egingo dugu atera daitezkeen ondorioen semantika handiagoa lortu ahal izateko. Lehenengo, J48Consolidated-en emaitzetarako bakarrik, aztertuko dugu ea estaldura maila handitzeak merezi duen. Gero aztertuko ditugu laginketa mota desberdinen arteko emaitzak, baina jada lehiakorra den estaldura maila batentzat eta funtsa desberdina duten bi laginketa multzotan banatuta: lagin estratifikatuak eta *bootstrap* laginak, alde batetik multzo berean, eta orekatuak diren 2 laginketak beste multzo batean. Azkenik, J48Consolidated-en oinarria den J48rekin konparatuko ditugu bi laginketa multzoetatik J48Consolidated-en ordezkari hoberenak.

Gehiegi ez luzatze aldera, hona hemen azterketa hierarkiko honen gako nagusiak (kasu honetan, beti 36 datu-baseekin egina):

— *Estaldura mailan oinarritutako azterketa*. Lehenengo J48Consolidated-en emaitzen diferentziak aztertuko ditugu baina laginketa mota bereko estaldura maila desberdinetarako. Beraz, 7 azterketa independente izango ditugu, bat laginketa mota bakoitzeko. Aipatu denez, 11 estaldura balio desberdin erabili ditugu; haien arteko diferentzien azterketa egiteko, berriz, lagin kopurua hiru den estaldura baliorik altuenetik aurrera dauden estaldura maila desberdinak bakarrik hartu dira kontuan, lagin kopurua hiru duten estaldura mailak saihestuz. %90eko tamainako lagin estratifikatuak erabiltzen ditugunean, adibidez, bi estaldura maila diferente bakarrik geratzen zaizkigu, %99koa (3 laginekin jadanik lortzen da) eta %99,9koa (4 laginekin). Kasu honetan, Wilcoxon da erabili beharreko testa. Gainerako guztietan Friedman-ena.

Friedman-en arabera, laginketa mota guztietarako, «estratifikatua %tam=90» kasuan izan ezik, estaldura maila desberdinen artean diferentzia esanguratsuak daude beti (0,05eko maila edo atalasea jarrita). Kasu guztietan, jakina, estaldura mailarik altuenen alde: beti %90eko estaldura mailatik aurrera. Ranking-eko lehenengo posizioa laginketa gehiagotan lortu duten estaldura mailak bi balio altuenak izan dira: %99 eta %99,9. Bien artean diferentzia esanguratsurik inoiz agertu ez denez gero, %99ko estaldura maila hartu dugu ordezkari gisa, lagin multzoa txikiagoa —eta beraz zuhaitzak sinpleagoak— izate aldera.

— *Laginketa moten arteko azterketa*. Behin %99ko estaldura maila ezarrita, laginketa mota desberdinen artean zuhaitz kontsolidatuek eman ditzaketen emaitzak aztertzeari ekingo diogu. Laginketa prozesuaren izaera oso desberdina denez, bi multzotan banatuko dugu haien arteko konparazioa: alde batetik, lagin estratifikatuen (*bootstrap*-ekoak barne) artean eta, bestetik, orekatuen artean, multzo bakoitzean egokiena zein den baloratzeko.

Bi azterketatan ere diferentzia esanguratsuak aurkitu ditugu. Lagin estratifikatuen kasuan, Friedman aplikatuta, honakoa izango litzateke ordena, hoberenetik okerrenera (%tam adierazita): %75, %50, %90, *bootstrap* eta, azkenik, %25. Bestalde, lagin orekatuen kasuan, Wilcoxon erabilita (bi alternatiba izaki), tamaina maximoko laginak (*tamMax*) erabiltzen dituzten zuhaitz kontsolidatuen alde ateratzen dira diferentziak (aurreko lanekin [16] bat etorrita).

— *J48Consolidated-en ordezkarien eta J48ren arteko azterketa*. Azkenik, J48Consolidated-en bi ordezkarien («estratifikatuak %tam=75» eta «orekatuak *tamMax*», %99ko estaldura maila erabilita) eta J48ren arteko konparazioan, Friedman erabilita, diferentzia estatis-

tikoki esanguratsuak daudela suertatzen da, *p-value* 0,0162 izanik. Lagin estratifikatuak erabiltzen dituen J48Consolidated-ek eskuratzen du rankingik hoberena (1,67), bigarrena orekatuenak (2,0) eta, azkena, J48 sailkatzaileak (2,33 rankingarekin). Gainera, *post-hoc* moduko Shaffer testaren arabera, lehenengo (J48-Consolidated lagin estratifikatuekin) eta azkeneko (J48) aukeren artean diferentzia esanguratsuak eskuratzen dira (*p-value* a 0,0140 izanik) eta ez ordea, lehenengo bi aukeren artean (bi J48Consolidated-enak).

5. **ONDORIOAK**

Zuhaitz kontsolidatuak sortzeko algoritmoa, *CTC*, diseinatu zen lagin multzo batean dagoen ezagutza guztia sailkatzaile batean jasotzeko, sailkapenaren azalpenik galdu gabe. Artikulu honetan CTC algoritmoaren WEKA plataformarako inplementazioa, *J48Consolidated* paketea alegia, aurkeztu da. J48Consolidated-en urteetan zehar CTCrekin erabili izan diren hainbat laginketa mota desberdin inplementatu dira eta, bide batez, berriki argitara eman den *estaldura*n oinarritutako teknika bat ere integratu da. Honen bidez, jatorrizko laginaren kasuen estaldura minimo bat bermatu daiteke lagin multzoan, informazioa gal ez dadin. Bide batez, J48Consolidated-en oinarrizko erabilpena WEKAko tresna nagusienetan azaldu da.

Gero, J48Consolidated sailkatzaile moduan proban jartzeko, esperimentu zabal bat egin da laginketa desberdinek sailkatzeko garaian duten eragina aztertzeko. Esperimentu hau *UCI* biltegiko 36 sailkapen-problemen gainean burutu da, guztira 7 laginketa mota desberdin erabilita (*bootstrap* laginak, 4 tamaina desberdinetako lagin estratifikatuak eta 2 tamainako lagin orekatuak). Gainera, estaldurak duen eragina aztertu da laginketa hauetan, oso balio baxuetatik abiatuta %99,9ko estaldura mailaraino. J48 sailkatzailea, J48Consolidated-en oinarrituta dagoen erabaki-zuhaitz arrunta, ere gehitu da konparazioan, oinarriko erreferente gisa.

Emaitzak erakusteko garaian, diferentzia estatistikoki esanguratsuen azterketa hierarkiko batean oinarritzea aukeratu da. Honela, lehenengo azterketaren arabera merezi du balio altuko estaldura mailak erabiltzea laginketa guztietan. Beraz, %99ko estaldura maila erabiltzea erabaki dugu. Lagin estratifikatuak eta *bootstrap* laginak erabiltzen ditugunean, diferentzia esanguratsuak lortzen dira %75eko lagin estratifikatuak erabiliz gero. Bestalde, lagin orekatuak erabiltzen ditugunean, azpilagindu daitekeen tamaina maximoarekin lortzen dira diferentzia esanguratsuak. Azkenik, bi lagin mota hauekin sortutako J48Consolidated zuhaitzak J48koekin konparatzen ditugunean, lagin estratifikatuko zuhaitzek lortzen dituzte emaitzarik hoberenak, lagin orekatuen emaitzak datoz gero eta J48ko zuhaitzenak azkenik.

Gainera diferentzia esanguratsuak daude J48Consolidated-en lehenengo aukerarekin alderatuta. Beraz, %75eko tamainako lagin estratifikatuekin eraikitako J48Consolidated zuhaitz kontsolidatuak erabiltzea oso aukera lehiakorra bilakatzen da, oro har, edozein motatako sailkapen-problema ebatzi nahi dugunean.

Epe motz eta ertain batean, gaur egun aurreratuagoak diren beste laginketa mota batzuk ere (SMOTE bezalako birlaginketa adimentsua [28], adibidez) inplementatzea eta J48Consolidated sailkatzailean integratzea izango litzateke gure helburutako bat. Bestalde, sailkapen-algoritmoen ikuspuntutik, oso interesgarritzat jotzen dugu kontsolidazio prozesua beste erabaki-zuhaitzeko algoritmoetan (CART, CHAID... bezalakoak) aplikatzea eta WEKAn ere gehitzea, edo, zergatik ez, baita sailkapenaren azalpena eskaintzen duten erregela-multzoko algoritmoetan ere (Ripper edo PART, adibidez).

6. **BIBLIOGRAFIA**

[1] KASS G.V. 1980. «An exploratory technique for investigating large quantities of categorical data», *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **29(2)**, 119-127.

[2] MAGIDSON J. 1993. *SPSS for Windows: CHAID, Release 6.0*. SPSS Incorporated.

[3] BREIMAN L., FRIEDMAN J.H., OLSHEN R.A. eta STONE C.J. 1984. *Classification and Regression Trees*. Wadsworth.

[4] QUINLAN J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[5] QUINLAN J.R. 1986. «Induction of decision trees>, *Machine Learning*, **1(1)**, 81-106.

[6] WU X., KUMAR V., QUINLAN J.R., GHOSH J., YANG Q., MOTODA H., MC-LACHLAN G.J., NG A., LIU B., YU P.S., ZHOU Z.H., STEINBACH M., HAND D.J. eta STEINBERG D. 2008. «Top 10 algorithms in data mining», *Knowledge and Information Systems*, **14(1)**, 1-37.

[7] BREIMAN L. 1996. «Bagging predictors», *Machine Learning*, **24(2)**, 123-140.

[8] FREUND Y. eta SCHAPIRE R.E. 1996, «Experiments with a new boosting algorithm», *Proceedings of the Thirteenth International Conference on Machine Learning*, 148-156.

[9] BREIMAN L. 2001. «Random Forests», *Machine Learning*, **45(1)**, 5-32.

[10] BANFIELD R.E., HALL L.O., BOWYER K.W. eta KEGELMEYER W.P. 2007. «A Comparison of Decision Tree Ensemble Creation Techniques», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29(1)**, 173-180.

[11] DOMINGOS P. 1997. «Knowledge acquisition from examples via multiple models», *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*. Morgan Kaufmann, 98-106.

[12] PÉREZ J.M., MUGUERZA J., ARBELAITZ O. eta GURRUTXAGA I. 2004. «A new algorithm to build consolidated trees: study of the error rate and steadiness», *Advances in Soft Computing. Proceedings of the International Intelligent Information Processing and Web Mining Conference (IIS: IIPWM04)*, 79-88.

[13] PÉREZ J.M., MUGUERZA J., ARBELAITZ O., GURRUTXAGA I. eta MARTÍN J.I. 2007. «Combining multiple class distribution modified subsamples in a single tree», *Pattern Recognition Letters*, **28(4)**, 414-422.

[14] HALL M., FRANK E., HOLMES G., PFAHRINGER B., REUTEMANN P. eta WITTEN I.H. 2009. «The weka data mining software: an update», *SIGKDD Explorations Newsletter*, **11**, 10-18.

[15] ARBELAITZ O., GURRUTXAGA I., LOZANO F., MUGUERZA J. eta PÉREZ J.M. 2013. «J48Consolidated: An implementation of CTC algorithm for WEKA», University of the Basque Country (UPV/EHU), Technical Report EHU-KAT-IK-05-13, 1-34.

[16] IBARGUREN I., PÉREZ J.M., MUGUERZA J., GURRUTXAGA I. eta ARBELAITZ O. 2015. «Coverage-based resampling: Building robust consolidated decision trees», *Knowledge-Based Systems*, **79**, 51-67.

[17] IBARGUREN I., PÉREZ J.M., MUGUERZA J., ARBELAITZ O. eta GURRUTXAGA I. 2014. «An update of the J48Consolidated WEKAs class: CTC algorithm enhanced with the notion of coverage», University of the Basque Country (UPV/EHU), Technical Report EHU-KAT-IK-02-14, 1-48.

[18] ALBISUA I., ARBELAITZ O., GURRUTXAGA I., MUGUERZA J., PÉREZ J.M. eta PERONA I. 2010. «Obtaining optimal class distribution for decision trees: comparative analysis of CTC and C4.5», *Lecture Notes in Computer Science. Current Topics in Artificial Intelligence, CAEPIA 2009 Selected Papers. Springer-Verlag*. P. Meseguer, L. Mandow and R.M. Gasca (Eds.), **5988**, 101-110.

[19] ALBISUA I., ARBELAITZ O., GURRUTXAGA I., LASARGUREN A., MUGUERZA J. eta PÉREZ J.M. 2012. «Analysis of the effect of changes in class distribution in C4.5 and consolidated C4.5 tree learners», University of the Basque Country (UPV/EHU), Technical Report EHU-KAT-IK-01-12, 1-80.

[20] ALBISUA I., LASARGUREN A., MUGUERZA J. eta PÉREZ J.M. 2012. «Datuen birlaginketa adimentsua ikasketa automatikoan», *Ekaia*, **25**, 217-234.

[21] PÉREZ J.M., ALBISUA I., ARBELAITZ O., GURRUTXAGA I., MARTÍN J.I., MUGUERZA J. eta PERONA I. 2010. «Consolidated trees versus bagging when explanation is required», *Computing*, **89**, 113-145.

[22] BAUER E. eta KOHAVI R. 1999. «An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants», *Machine Learning*, **36**, 105-139.

[23] LICHMAN M. 2013. *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science.

[24] HUANG J. eta LING C. 2005. «Using AUC and accuracy in evaluating learning algorithms», *IEEE Transactions on Knowledge and Data Engineering*, **17**, 299-310.

[25] FAWCETT T. 2006. «An introduction to ROC analysis», *Pattern Recognition Letters*, **27**, 861-874.

[26] DEMŠAR J. 2006. «Statistical comparisons of classifiers over multiple data sets»>, *Journal of Machine Learning Research*, **7**, 1-30.

[27] GARCÍA S. eta HERRERA F. 2008. «An extension on «Statistical comparisons of classifiers over multiple data sets» for all pairwise comparisons», *Journal of Machine Learning Research*, **9**, 2677-2694.

[28] CHAWLA N.V., BOWYER K.W., HALL L.O. eta KEGELMEYER W.P. 2002. «SMOTE: Synthetic Minority Over-sampling Technique», *Journal of Artificial Intelligence Research*, **16**, 321-357.