eman ta zabal zazu

UPV EHU

ZIENTZIA ETA TEKNOLOGIA FAKULTATEA

FACULTAD DE CIENCIA Y TECNOLOGÍA

50 URTE AÑOS 1968 - 2018

Biba Zientzia! Ciencia Viva

# Machine Learning bidezko mezu toxikoen detekzioa

Gradu Amaierako Lana
Matematikako Gradua

Ane Acha Gonzalez

Mikel Penagarikano
Irakasleak zuzendutako lana

Leioa, 2024ko ekainaren 21a

# Gaien Aurkibidea

# Sarrera

Gaur egun, sare sozialek izugarrizko eragina dute gure egunerokoan eta eztabaidaezina da pertsonen arteko komunikazio modua aldatu egin dutela. Arestian, gazteen artean elkarrizketak mantentzeko tokia zela pentsatu izan den arren, zalantzarik gabe sare sozialak gizarte osorako eta arlo guztietara hedatu dira, komunikabide soil bat baino gehiago bihurtuz. Orain, harreman pertsonaletan eta sozialetan ez ezik, negozioak egiteko moduan ere eragiten dute. Egunero erabiltzen ditugun tresnak dira, elkarri eragiteko edo eduki pertsonal eta profesionala sortzeko.

Beraz, gizartean ardura handiko gaia dela ikusita, ezinbestekoa da sarean argitaratutakoa kontrolpean izatea. Batez ere, mezu arrazistak, bortitzak... detektatzea beharrezkoa da, komunikazioa egokia izan dadin.

Horretarako, lan honetan, sare sozial zehatz bateko mezuen sailkapen automatikoa landu da. Hau da, Machine Learning metodoen bidez mezu toxikoak detektatu ditugu.

Machine Learning-a, adimen artifizialaren azpidiziplina, ikasteko gaitasuna duten sistemak sortzeagatik bereizten da [1][2]. Adar honek, aztertutako eta prozesatutako datuetan oinarrituz, bere kabuz ikasi dezaketen sistemak sortzeko balio du. Algoritmo desberdinen bitartez, ordenagailuek patroiak identifikatzeko aukera dute, ondoren predikzioak egin ditzaten.

Lan honek Machine Learning-aren zenbait klasifikazio eredu frogatzea eta alderatzea du helburu, gure datuetan aplikatu ahal izateko eta mezu toxikoak iragazteko.

Lana honako eran antolatuta dago:

1. kapituluan, aurre egingo diogun problemaren azalpena eta aukeratutako datu basearen analisia aurkezten dira. Gainera, lanaren zehar erabilitako softwareari buruzko azalpen txikia ematen da bertan, eta ereduak ebaluatzeko jarraituko diren metrikak azaltzen dira.

Hurrengoko kapituluetan erabili ditugun ereduak azaltzen dira. Machine Learning algoritmo guztien artean bi modalitate nagusi bereiz ditzakegu datuen arabera: ikaskuntza gainbegiratua eta gainbegiratu gabea [3]. Gure datuak aldez aurretik etiketatuak daudenez, ikaskuntza gainbegiratuaren algoritmoekin arituko gara lanean. Haien artean honakoak landu ditugu: Naive Bayes (2. kapitulua eta baliokidea den bektoreen bidezko eredua, 3. kapitulua), Erregresio Logistikoa (4. kapitulua), Support Vector Machine (5. kapitulua) eta Decision Tree eta Random Forest (6. kapitulua). Kapitulu horietan lehenik algoritmoak matematikoki ulertzea izango dugu helburu eta ondoren, ereduak entrenatu eta ebaluatuko dira.

7. kapituluan, bildutako informazio guztia erabiliz ateratako ondorioak irakur daitezke. Amaitzeko, erabilitako bibliografia eskaintzen da eta, azkenik, eranskinetan, lagungarria izan daitekeen optimizazio teoria eta ereduak programatzeko garatu den kodea dago.

# 1. Kapitulua

# Problemaren eta datu basearen azalpena

## 1.1 Problemaren azalpena

Azken urteetan, Machine Learning-aren arloan, erronka handienetariko bat testuekin lan egitea da. Testua klasifikatzeko ereduak gero eta indar handiagoa hartzen ari dira, denbora laburrean algoritmo berri dezente sortu direlarik.

Lan hau mezuak toxikoak diren edo ez sailkatzean datza. Horretarako, Kaggle [4] plataformak eskaintzen duen datu base batekin arituko gara lanean. Kaggle datu zientzia arloko plataforma handienetariko bat da, erabiltzaileei datu multzoak eskaini eta haiekin jarduteko aukera ematen diena. Horrez gain, erronkak proposatzeko oso erabilia da, bertan enpresek zein Kaggle-k berak ebazteko problema ezberdinak planteatzen dituzte, gehienetan sari baten truke, eta erabiltzaileek haien emaitzak igo ditzakete[5].

Bertan agertzen diren erronka guztien artean, lan honetan erabiliko dugun datu basea dago eskuragarri, hain zuzen ere, *Quora Insincere Questions Classification* [6].

Problema honetan, Quora sare sozialean egindako galderak sailkatuko ditugu. Quoran, jendeak galderak egiteko eta erantzuteko aukera du, baina, beste webguneetan ez bezala, benetako eta kalitatezko informazioa partekatzen dela ziurtatzen da. Hori dela eta, plataforma honen helburu nagusienetariko bat mezu toxikoak desageraraztea da, jakintza sustatzeko konfiantzazko gunea izaten jarrai dezan eta erabiltzaileak seguru sentitu daitezen [7].

Ondoren aurkezten diren ezaugarriek mezu toxiko baten adierazle izan daitezke:

- Tonu ez neutrala: Neurriz kanpoko tonuaren erabilera iritzia azpimarratzeko, galdera erretorikoen erabilera...
- Mezu probokatzaileak edo mespretxuzkoak: Klase-maila desberdinen arteko ideia diskriminatzaileak esatea, estereotipo baten edo pertsonatalde jakin bati eraso edo irain egitea...
- Informazio faltsua edo zentzugabea partekatzea.
- Eduki sexuala erabiltzea.

## 1.2 Softwarea

Lana *Python* programazio lengoaiaren bidez egin dugu. Python goi-mailako programazio lengoai interpretatua da [8]. Haren ezaugarri nagusia sintaxi garbia, eraginkorra eta irakurtzeko erraza duela da.

Pythonek duen oinarrizko ezaugarri bat bere liburutegiak dira, izan ere, hizkuntzaren oinarrizko funtzionaltasuna areagotzen duten klase eta metodoak eskuratzeko aukera ematen dute. Erabili ditugun liburutegi esanguratsuenak hauek dira:

- **Pandas:** Liburutegi hau datuen analisirako prestatuta dago, datuekin era eraginkorrean lan egitea ahalbidetzen duelako. Datuak eraldatzeko, bilaketa bizkorrak egiteko, birordenatu, banatu, datuekin lan egiteko egitura flexibleak eskaintzen ditu. Ildo beretik, fitxategiak kargatzeko oso erabilgarria da (*.xlsx*, *.csv* zein *.txt* formatukoak) [9].
- **NumPy:** Zenbakizko kalkuluan eta analisian espezializatutako liburutegia da. Objektu klase berri bat sortzen du, 'array' izena hartzen duena, datu bildumak zenbait dimentsiotan adieraztea ahalbidetzen duena. 'array'-ak zerrendak baino bizkorragoak dira, hortaz, hobeagoak dira matrizeekin zein bektoreekin lan egiterakoan [10].
- **NLTK** (Natural Language Tool Kit): Hizkuntzaren prozesamendurako NLTK plataforma liderra da. Interfaze erabilerrazak eskaintzen ditu 50 corpus eta baliabide lexikal baino gehiagotara. Horrekin batera, testuak prozesatzeko liburutegi multzo bat eskaintzen du, sailkapen, tokenizazio eta stemming aukerekin, besteak beste [11][12].
- **Pickle:** Objektuak fitxategi bitarretan erraz gordetzeko aukera ematen digu. Liburutegi honek Python-eko objektuak byte segida bihurtzen ditu, informazioa fitxategi bitarretan era bizkorrean gordetzeko [13].

- **Gensim:** Dokumentuak bektore semantiko gisa errepresentatzen ditu, modurik eraginkorrenean eta sinplenean irudikatzeko [14].
- **Scikit-learn:** Python-eko liburutegi honek gainbegiratutako eta gainbegiratu gabeko ikaskuntza-algoritmo ugari eskaintzen ditu, hala nola, erregresio logistikoa, support vector machine eta decision tree. Tresna hau oso erabilgarria da datu-zientzian eta ikaskuntza automatikoan, hainbat sailkapen, erregresio eta taldekatze-algoritmo eskaintzen dituelako, NumPy, Pandas eta Matplotliben gainean eraikita dago [15].
- **Matplotlib:** Grafikoak irudikatzen ditu, hala nola, histogramak eta zuzen diagramak, besteak beste [16].

## 1.3 Datu basearen azalpena

1.1. irudiak datu basearen lehenengo eta azkeneko bost elementuak erakusten ditu eta hurrengo argumentuak hartzen dituela ikus dezakegu:

- quid : erabiltzailea.
- questiontext : idatzitako galdera (mezua).
- target : 0 (mezu ez-toxikoa) edo 1 (mezu toxikoa).

| | qid | question_text | target |
|---|---|---|---|
| **0** | 00002165364db923c7e6 | How did Quebec nationalists see their province... | 0 |
| **1** | 000032939017120e6e44 | Do you have an adopted dog, how would you enco... | 0 |
| **2** | 0000412ca6e4628ce2cf | Why does velocity affect time? Does velocity a... | 0 |
| **3** | 000042bf85aa498cd78e | How did Otto von Guericke used the Magdeburg h... | 0 |
| **4** | 0000455dfa3e01eae3af | Can I convert montra helicon D to a mountain b... | 0 |
| **...** | ... | ... | ... |
| **1306117** | ffffcc4e2331aaf1e41e | What other technical skills do you need as a c... | 0 |
| **1306118** | ffffd431801e5a2f4861 | Does MS in ECE have good job prospects in USA ... | 0 |
| **1306119** | ffffd48fb36b63db010c | Is foam insulation toxic? | 0 |
| **1306120** | ffffec519fa37cf60c78 | How can one start a research project based on ... | 0 |
| **1306121** | ffffed09fedb5088744a | Who wins in a battle between a Wolverine and a... | 0 |

**1.1. Irudia**: Datu basea.

Ondorioz, gure betebeharra mezuak toxikoak diren edo ez antzematea denez, gure erantzun aldagaia 'target' izango da. Hau da, sortutako ereduek iragarri beharko duten datua.

## 1.4 Datuen analisia

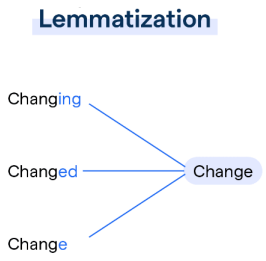
Eredu bat planteatu aurretik, ezinbestekoa da datuak prestatzea eta txukuntzea, makinak datuak analizatu ahal izateko. Datu gordinak eraldatu behar dira, ereduak datuen ezaugarriak erraz antzeman ditzan. Oro har, iturri ezberdinetatik hartutako datuek kalitate-arazoak izan ditzakete, hala nola, balio faltak, balio errepikatuak edo eskuzko laginketa okerrak. Arazo horiek ekiditeko eta datu erabilgarriak ezabatzeko prozesuari datuen aurreprozesaketa deritzo.[17]

Hasteko, testuarekin lan egiterako orduan, funtsezko teknika tokenizazioa da. Hau da, testua token deritzoten zati txikiagotan zatitzeko prozesua. Token horiek banakako hitzak, zenbakiak, puntuazio-markak edo testuaren beste unitate esanguratsu batzuk izan daitezke.

Hori lortzeko *gensim.utils.simple_preprocess* [18] funtzioa erabili dugu. Funtzio honek zerrrenda batean bueltatzen du testu tokenizatuta. Ez hori bakarrik, defektuz zenbait zeregin gehiago egiten ditu, esate baterako, guztia minuskuletan adierazi eta puntuazio-markak eta zenbakiak ezabatu, besteak beste.

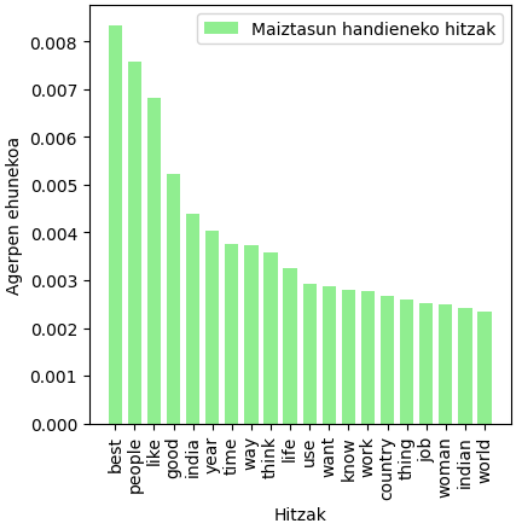
Testua garbitzeko ohikoak izaten diren beste bi teknika ere egin ditugu. Alde batetik, StopWords hitzak ezabatzea. Stop Words (hitz hutsak) esanahirik ez duten hitzak dira, esate baterako, artikuluak, izenordainak, preposizioak... Horrelako hainbat hitz ditugu mezuetan eta ez dutenez informazio handirik ematen mezuak sailkatzerako orduan, ezabatu egin ditugu.

Beste aldetik, lematizazioa erabili dugu. Honek hitzak beraien forma kanonikora edo haien oinarrira bueltatzen ditu. Prozesu honen abantaila nagusia hitz ezberdin kantitatea (lexikoa) asko murrizten duela da, baina ia informaziorik galdu gabe. Hona hemen adibide sinple bat:

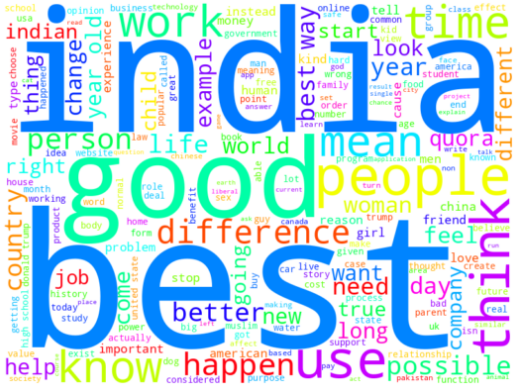



**1.2. Irudia**: Lematizazioaren adibidea.

Behin datuen garbiketa eginda dagoela, mezuetan maiztasun handienekin agertzen diren hitzak erakutsiko ditugu. 1.3. barra diagraman ikus daitezke best, people eta like direla, hurrenez hurren. Bukatzeko, era bisualago batean adierazteko hitz hodeia (Word Cloud, ingelesez) irudikatu dugu. Hitzen irudikapen bat da non hitz bakoitzaren tamainak datuen multzoan duen maiztasuna adierazten duen [19], ikusi 1.4. irudia.




**1.3. Irudia**: Barra diagrama.




**1.4. Irudia**: Hitz hodeia.

Amaitzeko, aurretik aipatu bezala, gure erantzun aldagaia etiketak izango dira: mezu ez toxikoa (0 klasea) edo toxikoa (1 klasea). Datu baseko klaseak oso desorekatuak daude, izan ere, mezu guztietatik, 1225312 (% 94) ez-toxikoak dira eta 80810 (% 6) bakarrik dira toxikoak. Erantzun aldagaiak banaketa binomiala jarraitzen du: $Y : Bin(1, 0.06)$.

## 1.5 Datuen banaketa

Datuak prestatuta daudela, datu multzoa bitan (train eta test) edo hirutan (train, validation eta test) banatu ohi da. Gure kasuan, datu multzoen banaketa ondorengoa izango da: % 80 entrenamendua (train), % 10 balidazioa (validation) eta % 10 testa (test). Ereduak sortzeko entrenamenduko datu multzoak erabiliko ditugu eta ereduak ebaluatzeko, aldiz, balidazio eta test multzoak [20].

## 1.6 Ebaluazioa

Lan osoan zehar egingo ditugun ereduak ebaluatzeko, jarraian azaltzen diren metrikak erabiliko ditugu eta horien bitartez ereduak konparatzeko eta ondorioak lortzeko aukera izango dugu.

#### Sailkapen Zuzeneko Probabilitatea

Gehien erabiltzen den tresnetako bat sailkatze taula da, intepretatzeko erreza baita. $2 \times 2$ -ko taula da, Y -ren behatutako eta estimatutako balioekin osatzen dena. Hau da, estimatutako probabilitatea aukeratutako c mozketa puntua baino handiagoa bada, orduan indibiduo hori arrakasta bezala sailkatua izango da, eta bestela, porrot bezala. Honakoa da sailkatze taula:

$$\hat{y}_i = \begin{cases} 1 & \hat{p}_i \geq \theta \\ 0 & \hat{p}_i < \theta \end{cases} \quad \forall i = 1, \ldots, n$$

| | **Estimatutakoa** ($\theta$) | |
|---|---|---|
| **Behatutakoa** | $\hat{y} = 0$ | $\hat{y} = 1$ |
| $y = 0$ | EN | PF |
| $y = 1$ | NF | EP |

**1.1. Taula**: Sailkatze taula

- Egiazko Negatibo (EN) : Ereduak porrot bezala egoki sailkatu dituen porrot kopurua da.
- Positibo Faltsuak (PF) : Ereduak arrakasta bezala desegoki sailkatu dituen porrot kopurua da.
- Negatibo Faltsuak (NF) : Ereduak porrot bezala desegoki sailkatu dituen arrakasta kopurua da.
- Egiazko Positiboak (EP) : Ereduak arrakasta bezala egoki sailkatu dituen arrakasta kopurua da.

Horiek erabilita, hurrengo metrikak erabiliko ditugu gure ereduaren doikuntza neurtzeko:

- Zehaztasuna (Accuracy): Egoki sailkatutako proportzio globala adierazten du.

$$Zehaztasuna(\theta) = \frac{EP + EN}{EP + EN + NF + PF}$$

- Prezisioa (Precision): Ereduak arrakasta bezala sailkatu dituenetatik, zenbat diren benetan arrakasta adierazten du.

$$Prezisioa(\theta) = \frac{EP}{EP + PF}$$

- Sentikortasuna (Recall) :Ereduak benetan arrakasta diren kasuak antzemateko duen gaitasuna adierazten du. Hau da, benetako arrakasta diren kasuetatik zenbat ondo detektatzen dituen neurtzen du:

$$Sentikortasuna(\theta) = \frac{EP}{EP + NF}$$

- Espezifikotasuna (Specificity): Ereduak benetan porrota diren kasuak antzemateko duen gaitasuna adierazten du. Hau da, benetako porrota diren kasuetatik zenbat ondo detektatzen dituen neurtzen du.

$$Espezifikotasuna(\theta) = \frac{EN}{EN + PF}$$

**AUC - Area Under the Curve**

Mozketa puntu desberdinentzako, emaitzak denak batera kontuan hartzen dituen parametro bat AUC, Area Under the Curve, da. AUC-ak doitutako ereduak duen aurresateko gaitasuna neurtzen duen estatistikoa da.

c mozketa puntuaren arabera lortutako eredu baten Sensibilitatea eta (1-Espezifizitatea) elkarrekin irudikatzen dituen grafikoari ROC kurba (Receiver Operating Characteristic) deritzo [21]. ROC kurbaren azpian geratzen den azalera da hain zuzen AUC estatistikoa [22]. AUC zenbat eta gehiago hurbildu 1era hobea izango da ereduaren aurresateko gaitasuna.

Aldiz, Precision-Recall Curve, Prezisioa eta Sensibilitatea elkar irudikatzen ditu. ROC kurba bezala, c mozketa puntu desberdinetarako hartzen ditu balioak eta kurba honetan ere AUC-a neur daiteke. Precision-Recall kurbak informazio gehiago ematen du klasifikazio bitarra eta klaseak desorekatuak direnean [23] [24]. Ondorioz, sortuko ditugun ereduetan erabakiak hartzeko eta ereduak elkar konparatzeko, Precision-Recall kurbaren AUC-an oinarrituko gara.

# 2. Kapitulua

# Naive Bayes

Eredu sinpleak batzuetan indartsuenak dira, eta Naive Bayes sailkatzailea horren adibidea da. Azken urteetan, Machine Learning arloan egin diren aurrerapenak izugarriak izan diren arren, klasifikatzaile zahar honek egiaztatu du, erraza izateaz gain, azkarra, eraginkorra eta fidagarria ere badela.

Naive Bayes klasifikatzaile probabilistikoa da, Bayesen teoreman oinarritzen dena [25][26]. Eredu honek duen berezitasun nagusia aldagaien arteko independentzia asumitzen duela da. Suposizio hau gehienetan aplikazio praktikoetan argi eta garbi hausten da eta, horregatik, *naive* (inozo) izena hartzen du.

## 2.1 Oinarri teorikoa

Hasteko, gogora dezagun Bayesen teorema:

**2.1.1. teorema. Bayesen Teorema.**
Izan bitez $(\Omega, \mathscr{F}, \mathcal{P})$ probabilitate-espazioa eta $(A_1, A_2, \ldots, A_n)$ gertaerak non $\dot{\bigcup}_{i=1}^{n} A_i = \Omega$ eta $P(A_i) \neq 0$ $i = 1, 2, \ldots, n$ diren. $B$ edozein gertaera dela jakinda non $P(B) \neq 0$ den, eta baldintzazko probabilitateak ezagunak izanik $(P(B|A_i))$, orduan, $A_i$ gertaeraren $B$-rekiko baldintzazko probabilitatea honela adieraz daiteke:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{2.1}$$

non $P(A_i)$ *a priori* probabilitateak, $P(A_i|B)$ *a posteriori* probabilitateak eta $P(B|A_i)$ $A_i$ jakinik $B$ gertatzeko probabilitateak diren.

Bayes-en teorema oinarritzat hartuz, eredu bat sor dezakegu. Izan bitez $Y$ erantzun aldagaia eta $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p)$ aldagai azaltzaileen multzoa. Suposatu $\mathbf{x}_i = (x_1, \ldots, x_n)$ dela $\forall \mathbf{x}_i \in \mathbf{X}$, $i = 1, \ldots, p$ eta $\mathbf{x}_i$ aldagai bakoitzari $y_j$ klase bat dagokiola, non $Y = \{y_1, \ldots, y_c\}$ den, $j = 1, \ldots, c$ izanik.

Naive Bayes klasifikatzaileak $P(Y = y_j \mid \mathbf{X} = \mathbf{x}_i)$ kalkulatzea du helburu. Hots, $\mathbf{x}_i$ aldagaia $y_j \in Y$ klasekoa izateko probabilitatea. Horretarako, Bayesen teorema besterik ez dugu aplikatu behar:

$$P(y_j \mid \mathbf{x}_i) = \frac{P(y_j)P(\mathbf{x}_i \mid y_j)}{P(\mathbf{x}_i)} \tag{2.2}$$

Aurretik esan bezala, eredu honek aldagaien arteko independentzia asumitzen duenez,

$$P(\mathbf{x}_i \mid y_j) = P(x_1, x_2, ..., x_n|y_j) = P(x_1|y_j)P(x_2|y_j)...P(x_n|y_j) \tag{2.3}$$

dugu eta ondorioz:

$$P(y_j|\mathbf{x}_i) = \frac{P(y_j)\prod_{k=1}^{n} P(x_k \mid y_j)}{P(\mathbf{x}_i)} \tag{2.4}$$

lortzen dugu. Ohartu $P(y_j \mid \mathbf{x}_i)$ balio maxioa bilatzean $P(\mathbf{x}_i)$ omititu dezakegula, ez baita $y_j$-ren menpekoa. Sailkatzerakoan, klase bakoitzerako zenbakitzailearen balioa kalkulatu eta balio hori maximoa duen klasea hautatuko dugu. Arau horri a posteriori maximoaren erregela *(maximum posterior rule)* deritzo. Hautatutako klaseari *maximum a posteriori (MAP)* esaten zaio, eta honela adierazten da klasearen estimatzailea:

$$\hat{y} = \underset{y_j \in Y}{\operatorname{argmax}} P(y_j \mid \mathbf{x}_i) = \underset{y_j \in Y}{\operatorname{argmax}} P(y_j) \prod_{k=1}^{n} P(x_k \mid y_j) \tag{2.5}$$

Bukatzeko, Naive Bayesen ohikoa izaten da logaritmoa aplikatzea, zenbakizko arazoak saihesteko. Baldintzapeko probabilitateak oso txikiak izan daitezke eta eremu logaritmikoan lan egiteak aukera ematen du probabilitate txiki horiek maneiatzeko, logaritmoak batuz, probabilitateak biderkatu beharrean. Ondorioz horrela geratuko litzateke:

$$\hat{y} = \underset{y_j \in Y}{\operatorname{argmax}} \log P(y_j \mid \mathbf{x}_i) = \underset{y_j \in Y}{\operatorname{argmax}} \log P(y_j) + \sum_{k=1}^{n} \log(P(x_k \mid y_j)) \tag{2.6}$$

## 2.2 Entrenamendua

Naive Bayes eredua testu-dokumentuen sailkapenerako erabiltzea ohikoa izaten da. Beraz, azaldu berri den teoria jarraituz, gure lehen eredua sortuko dugu. Horretarako, *train* datu multzoko mezuak erabiliko ditugu.

Hasteko, gure datu basean, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p)$ mezuen multzoa adierazten du, $p = 1044897$ izanik. Mezu bakoitza haren hitzen sekuentzia bezala definitu dezakegu, hau da, $\mathbf{x}_i = (x_1, \ldots, x_n)$ non $x_k$ $\mathbf{x}_i$ mezuaren hitza den, $k = 1, ...n$. $Y = \{0, 1\}$ erantzun aldagaia da, klaseen multzo bitarra adierazten duena, izan ere, mezua ez-toxikoa (0 klasea) edo toxikoa (1 klasea) izan daiteke.

Beraz, Naive Bayes algoritmoaren bitartez, edozein $\mathbf{x}_i = (x_1, \ldots, x_n)$ mezu ez-toxiko edo toxiko bezala sailakatuko dugu. Horretarako, 2.6 berdintzak adierazten duen moduan, $P(0 \mid \mathbf{x}_i)$ eta $P(1 \mid \mathbf{x}_i)$ probabilitate baldintzatuen logaritmoaren artetik baliorik handiena hautatuko dugu.

Lehenik eta behin, $P(x_k \mid 0)$ eta $P(x_k \mid 1)$ kalkulatuko ditugu, hau da, $x_k$ hitzaren probabilitatea klaseen arabera. Hori lortzeko, bi hiztegi sortu ditugu: bata 0 klaserako, , mezu ez-toxiko guztien hitzekin eta haien maiztasunekin, eta bestea 1 klaserako (toxikoak). Maiztasunek hitz bakoitzak klase horretan dituen agerpen erlatiboak adierazten dituzte, hots kalkulatu nahi dugun probabilitatearen hurbilpen enpirikoa.

Naive Bayesen oinarrizko ideia hitzak elkarren artean independienteak direla suposatzea da, nahiz eta errealitatean hitzen ordenak garrantzia izaten duten. Horrela, $\mathbf{x}_i$ mezuko hitz bakoitzaren maiztasunari logaritmoa aplikatu eta guztiak batzen baditugu, 2.6 berdintzaren eskuinaldeko bigarren terminoa lortzen dugu.

Kontuan izan behar dugu mezu bat aurresateko orduan, baliteke hiztegietan ez dagoen hitzaren bat agertzea. Hitz multzo honi 'OOV' (Out-Of-Vocabulary) izena emango diogu eta 0-ren ordez balio oso txiki bat esleituko diogu maiztasun bezala, biderkadura ez anulatzeko (batzen ditugun logaritmoak zentzua izateko). Balio hori nola zehaztu aurrerago azalduko dugu.

Beste alde batetik, klase bakoitzaren probabilitateak kalkulatu ditugu: $P(0)$ eta $P(1)$. Behin probabilitate kalkuluak eginda, mezua toxikoa den edo ez estimatuko dugu, 2.6 adierazten duen moduan. Hau da, 1 klasea (toxikoak) eta 0 klaseak (ez-toxikoak) antzeman dituzten bi probabilitateetatik handiena aukeratuz.

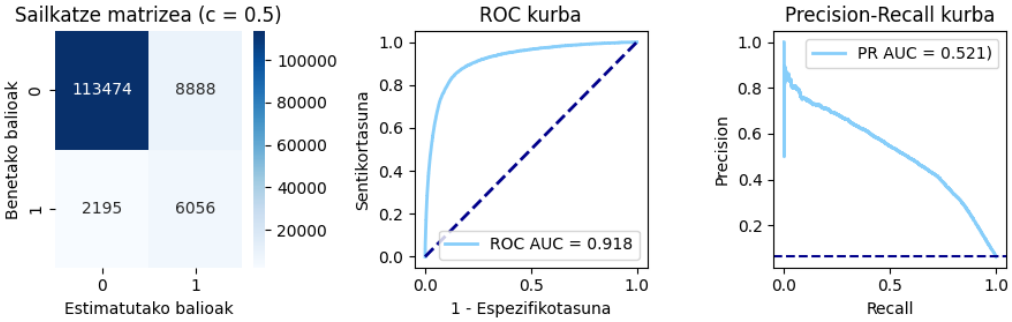
Bukatzeko, OOV hitzen maiztasun parametroa ezartzeko, balidazio azpimultzoan AUC irizpidea maximizatzen zuen balioa bilatu da, 0.1 balioa aukeratua izan delarik.

## 2.3 Ebaluazioa

Amaitzeko, ikus ditzagun lehen eredu honekin lortutako emaitzak.

2.1. irudian sailkatze matrizea, ROC kurba eta Precision-Recall kurbak irudikatzen dira, azken biak dagozkien AUC estatistikoarekin. Ereduaren zehaztasuna 0.92-koa da. Espezifikotasunak 0.93-ko balioa du eta, sentikortasunak, aldiz, 0.73-koa. Ondorioz, ez toxikoak (0 klasekoak) diren mezuak oso ondo sailkatu ditu,berriz, toxikoak (1 klasekoak) direnak ez hain ondo. Prezisioak balio txikiena du 0.41-koa.

ROC kurbaren AUC-a 0.92-koa da, beraz, ereduak aurresateko gaitasun altua duela adierazten du, 1-etik hurbil baitago. Aldiz, Precision-Recall kurbaren AUC-ari dagokionez, 0.52-ko balioa hartzen du. Grafikoan nabaria da ausazko eredu batekin alderatuta, gure eredua nahiko ona dela, baina AUC-ren balio horrek 1 klasearen aurresateko gaitasuna nahiko hobetu daitekeela adierazten digu.




**2.1. Irudia**: Naive Bayes ereduaren emaitzak.

Hurrengo taulan laburbiltzen dire aipatutako balio guztiak:

| ALGORITMOA | NAIVE BAYES |
|---|---|
| **PR_AUC** | **0.52** |
| ROC_AUC | 0.92 |
| Zehaztasuna | 0.92 |
| Prezisioa | 0.41 |
| Sentikortasuna | 0.73 |
| Espezifikotasuna | 0.93 |

**2.1. Taula**: Ereduen ebaluazio metriken laburpena.

Hurrengo ereduetan aldaketarik dagoen ikusiko dugu eta aurresateko gaitasuna hobea den edo ez.

# 3. Kapitulua

# Bektoreen bidezko eredua

Atal honetan, testuak bektore gisa nola errepresenta daitezkeen ikusiko dugu eta aurreko Naive Bayes ereduaren baliokidea den eredu berri bat aurkeztuko dugu, algebra linealari esker era oso laburrean adierazi ahal izango dena.

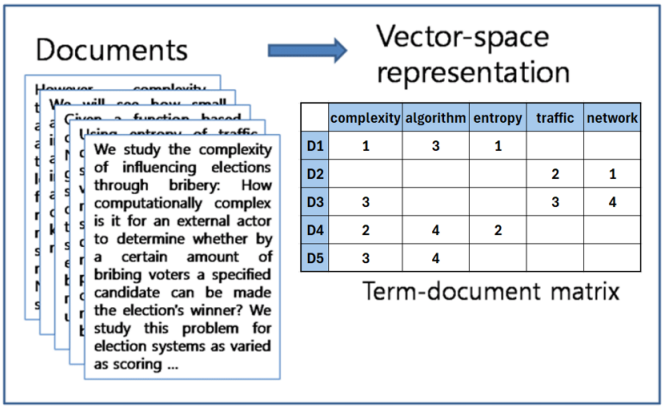
Gainera, testuaren bektorizazio hau hurrengo eredu guztietarako erabiliko dugu: erregresio logistikoa, Support Vector Machine, Decision Tree eta Random Forest.

## 3.1 Bektorizazioa

Espazio bektorialaren eredua (Vector Space Model) testu-dokumentuak bektore gisa irudikatzeko erabiltzen den eredu aljebraikoa da [27]. Normalean, informazioa berreskuratzeko, indexatzeko eta garrantziaren arabera sailkatzeko erabiltzen da.

Metodo honen oinarrizko idea dokumentu bakoitza bektore batean adieraztea da, lexikoko hitz bakoitzari bektorearen dimentsiotako bat dagokiolarik, eta, ondorioz, hitzen arteko erlazioaren konplexutasuna sinpleagoa bihurtuz.

Hona hemen adibide bat:




**3.1. Irudia**: Bektorizazioaren adibidea: matrizearen ilarak dokumentuak izango dira eta, zutabeak, berriz, hitzak. Zenbakiak dokumentu horretako hitzaren agerpen kopurua adierazten du.

Gure datuak horrela bektorizatuko ditugu:

$$M = \begin{pmatrix} m_{11} & \dots & m_{1q} \\ \vdots & \ddots & \vdots \\ m_{p1} & \dots & m_{pq} \end{pmatrix} \in \mathbb{N}^{p \times q} \tag{3.1}$$

non ilarak mezuak diren eta zutabeak hitzak, $p$ mezu kopurua izanik eta $q$ hitz desberdinen multzoa, hots lexikoa. Beraz, $m_{pq}$ balioak p. mezuan q. hitza zenbat aldiz dagoen adierazten du. q hitzak murriztuak direnez, azkenengo posizioa, OOV hitzentzat izango da (aurreko ereduan egin dugun bezala).

Bektorizatze prozesu hau programatzean, matrize sakabanatuak (sparse matrix) erabiltzea erabaki dugu, 0-z betetako matrizeak errepresentu eta haiekin kalkuluak egiteko eraginkorrak baitira [28]. Gure datu baseko mezu kopuru handiarekin, milioi bat mezu inguru, matrize dentsoa irudikatzea ezinezkoa litzateke. Matrize sakabanatuei esker, eraginkortasunez adieraz ditzakegu datu-egitura horiek, zero ez diren balioak eta haien kokapenak bakarrik biltegiratzen dituztelako eta gure kasuan, zero asko izango ditugulako (lexikoko hitz gehienak ez dira mezu bakoitzean agertuko). Hori dela eta, matrize hauek nabarmen murrizten dute karga konputazionala.

## 3.2 Eredu bektorialaren oinarriak

Bektoreen bidezko eredua planteatzeko, gure helburua aurreko kapituluko 2.6 berdintzaren baliokidea lortzea da, era bektorialean adierazita.

Hasteko, 2.6 berdintzako $P(\mathbf{x}_i \mid y)$ probabilitate baldintzatuaren logaritmoa hitzen agerpen kopuruaren eta maiztasunaren menpe idatz daiteke, izan ere:

$$\log P(\mathbf{x}_i \mid y_j) = \log \prod_{k=1}^{n} P(x_k \mid y_j) = \sum_{k=1}^{n} \log P(x_k \mid y_j) = \sum_{x \in \mathbf{x}_i} \#x \cdot \log P(x \mid y_j) \tag{3.2}$$

non $\#x$ $x$ hitzak $\mathbf{x}_i$ mezuan duen agerpen kopurua den eta $P(x \mid y_j)$ $x$ hitzaren maiztasunagatik hurbiltzen genuen balioa.

$\#x$ jadanik kalkulatu dugu, M agerpen matrizea hain zuzen ere.

Horrez gain, maiztasunen logaritmoak adierazten dituen V matrizea horrela definituko dugu:

$$V = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ \vdots & \vdots \\ v_{q1} & v_{q2} \end{pmatrix} = \begin{pmatrix} \log P(x_1 \mid 0) & \log P(x_1 \mid 1) \\ \log P(x_2 \mid 0) & \log P(x_2 \mid 1) \\ \vdots & \vdots \\ \log P(x_q \mid 0) & \log P(x_q \mid 1) \end{pmatrix} \tag{3.3}$$

V matrizearen lehenego zutabeak hitzen maiztasunaren logaritmoa adierazten du 0 klasean (ez-toxikoan), eta, bigarrenak, ordea, hitzen maiztasunaren logaritmoa 1 klasean (toxikoetan). Oraingoan ere, lexikoan agertu ez den hitzaren bat agertzen bada, 0.1eko maiztasun txikia esleituko diogu.

0 klasekoa eta 1 klasekoa izateko probabilitateen logaritmoak adierazten duen matrizea L bezala izendatzen badugu,

$$L = \begin{pmatrix} l_{11} & l_{12} \end{pmatrix} = \begin{pmatrix} \log P(0) & \log P(1) \end{pmatrix} \tag{3.4}$$

honakoa lortzen dugu:

$$A = M \cdot V + J_{p \times 1} \cdot L = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1q} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p1} & m_{p2} & \dots & m_{pq} \end{pmatrix} \times \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ \vdots & \vdots \\ v_{q1} & v_{q2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1_p \end{pmatrix} \cdot \begin{pmatrix} l_{11} & l_{12} \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{p1} & a_{p2} \end{pmatrix} = \begin{pmatrix} \log(P(\mathbf{x}_1 \mid 0)P(0)) & \log P((\mathbf{x}_1 \mid 1)P(1)) \\ \log(P(\mathbf{x}_2 \mid 0)P(0)) & \log P((\mathbf{x}_2 \mid 1)P(1)) \\ \vdots & \vdots \\ \log P((\mathbf{x}_p \mid 0)P(0)) & \log P((\mathbf{x}_p \mid 1)P(1)) \end{pmatrix} \tag{3.5}$$

non $a_{p1}$ $p$. mezua 0 klasekoa (ez-toxikoa) izateko probabilitatearen logaritmo desplazatua den eta, $a_{p2}$, aldiz, $p$. mezua 1 klasekoa (toxiko) izateko probabilitatearen logaritmo desplazatua den. Bi egoera posibleetatik balio handiena duen klasearekin gelditu ko gara, Naive Bayes ereduan azaldu genuen moduan.

Azkenik, emaitzei dagokienez, 2. kapituluan azaldu ditugunaren berdinak dira, 2.3. Ebaluazioa atalean ikus daitezke.

# 4. Kapitulua

# Erregresio Logistikoa

Erregresio metodoek, erantzun aldagai baten eta aldagai azaltzaile bat edo gehiagoren arteko erlazioa adierazten dute. Askotan, erantzun aldagaiaren banaketa binomiala izaten da eta, kasu horretan, erregresio logistikoa da analisia burutzeko metodo ohikoena.

Erregresio logistikoaren helburua zentzua duen doikuntza egokiena lortzea da, erantzun aldagaiaren eta aldagai azaltzaileen arteko erlazioa deskribatzeko. Bere berezitasuna erantzun aldagaia bitarra dela da [29][30].

## 4.1 Oinarri teorikoa

Izan bitez $Y$ banaketa binomialari darraion erantzun aldagai dikotomikoa eta $\mathbf{X} = \left(X_1, X_2, ..., X_p\right)^T$ non $X_1, X_2, ..., X_p$ $p$ aldagai aske diren. Izan bedi $P\left(Y = 1 \mid \mathbf{X}\right) = p\left(\mathbf{X}\right)$ arrakasta izatearen probabilitate baldintzatua. Erregresio logistikoaren eredua honako hau da:

$$p\left(\mathbf{X}\right) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \in (0, 1) \tag{4.1}$$

Hori kontuan izanda,

$$P\left(Y = 0 \,|\, \mathbf{X}\right) = 1 - P\left(Y = 1 \,|\, \mathbf{X}\right) = 1 - p\left(\mathbf{X}\right) = \frac{1}{1 + e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}} \tag{4.2}$$

*Logit transformazioa* deituko diogu $p\left(\mathbf{X}\right)$-ren transformazio honi:

$$g\left(p\left(\mathbf{X}\right)\right) = \mathit{logit}\left(p\left(\mathbf{X}\right)\right) = \ln\left(\frac{p\left(\mathbf{X}\right)}{1 - p\left(\mathbf{X}\right)}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \tag{4.3}$$

Ohartu logaritmoa aplika dezakegula bi probabilitate adierazpenak positiboak direlako, funtzio esponentzialak balio positiboak baino ez baititu hartzen, eta balio positiboen zatidura beti da positiboa.

*g esteka funtzioak* erregresio lineal orokorraren ereduaren propietate batzuk betetzen ditu: funtzio lineala da, jarraitua izan daiteke eta $(-\infty, \infty)$ balioak har ditzake.

Laginaren tamaina $n$ eta aldagai aske kopurua $p$ izanik, eredua matrizialki adierazita honakoa litzateke: $g\left(p\left(\mathbf{X}\right)\right) = \mathbf{X} \cdot \beta$ non

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \quad \text{eta} \quad \beta = (\beta_0, \beta_1, \beta_2, \dots, \beta_p)^T$$

diren.

**Parametroen estimazioa**

$\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^T$ parametroak estimatzeko *egiantz handieneko metodoa* erabiltzen da. Demagun $n$ tamainako lagin bat dugula eta izan bedi $(\mathbf{x}_i, y_i)$ $i = 1, \dots, n$ bikotea, non $y_i$ erantzun aldagaiaren balio dikotomikoa den eta $\mathbf{x}_i$ aldagai askeen bektorearen balioa i.gaian. Ondokoa da *egiantz funtzioa*:

$$l(\boldsymbol{\beta}) = \prod_{i=1}^{n} p(\mathbf{x}_i)^{y_i} [1 - p(\mathbf{x}_i)]^{1-y_i} \tag{4.4}$$

Izan bedi $L(\boldsymbol{\beta}) = \ln [l(\boldsymbol{\beta})]$. $L(\boldsymbol{\beta})$ maximizatzen duen $\boldsymbol{\beta}$ bilatzeko, $L(\boldsymbol{\beta})$ diferentziatuko dugu $\beta_j$ $j = 1, \dots, p$ bakoitzerako. Lortzen diren ekuazioei, *egiantz ekuazioak* deitzen zaie. $p + 1$ egiantz ekuazio lortuko ditugu guztira, $p + 1$ koefizienteei dagozkienak, hain zuzen ere. Honako hauek dira:

$$\frac{\partial L(\boldsymbol{\beta})}{\partial \beta_0} = \sum_{i=1}^{n} (y_i - p(\mathbf{x}_i)) = 0 \tag{4.5}$$

eta

$$\frac{\partial L(\boldsymbol{\beta})}{\partial \beta_j} = \sum_{i=1}^{n} x_{ij} (y_i - p(\mathbf{x}_i)) = 0 \quad \forall j = 1, \dots, p \tag{4.6}$$

Ekuazio hauek zenbakizko metodoak erabiliz ebatz daitezke, esate baterako, *Newton-Raphson*-en metodoaren bidez, ikusi [29].

**Parametroen interpretazioa**

Erregresio logistikoan, $\beta$ parametroak *odds ratio* kontzeptuak erabiliz interpretatzen dira. Aldagai azaltzaile eta arrakasta probabilitatearen arteko harremana ez-lineala denez, $\beta$ koefizienteak ez datoz bat X unitate

bat handitzearekin lotutako Y-ren probabilitatearen aldaketarekin.

*Odds*-a arrakastaren probabilitatearen eta porrotaren probabilitatearen arteko zatidura da, 0 eta $\infty$ artean edozein balio har dezake:

$$o = \frac{p(\mathbf{X})}{1 - p(\mathbf{X})} \tag{4.7}$$

non $p(\mathbf{X})$ arrakastaren probabilitatea adierazten duen parametroa den. *Odds ratioa (OR)* bi baldintzen arteko odds-ak alderatzen ditu eta $logit(p(\mathbf{X})) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ ereduaren koefizienteak interpretatzeko erabilia da.

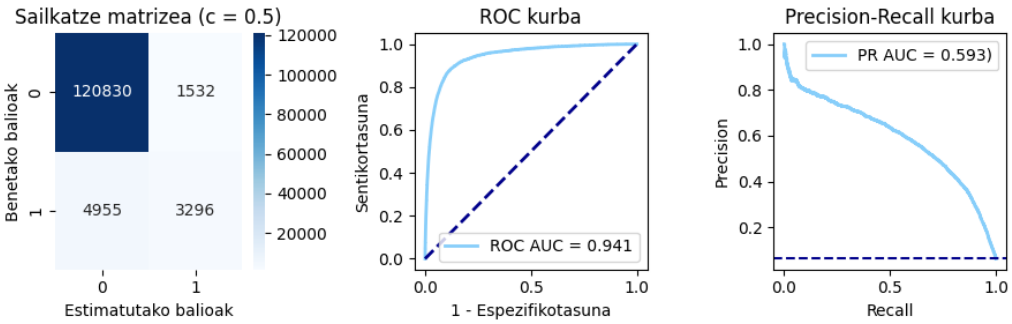
$\beta_j$ koefizientea $X_j$ aldagaiaren unitate bakoitzeko igoerak odds logaritmoan *(logit)* duen eragina adierazten du, beste aldagai guztiak konstante mantenduz. Koefiziente honen interpretazioa $e^{\beta_j}$ odds ratioaren bidez egiten da, eta horrek azaltzen du $X_j$-ren unitate bakoitzeko igoerak zenbat aldiz maizago emango den gertaera, gainontzeko aldagaiak konstante mantenduz. Hau da, $e^{\beta_j}$ odds ratioak adierazten du $X_j$ unitate bakoitzeko igoerak zenbat aldiz handitzen edo txikitzen dituen odds-ak.

## 4.2 Entrenamendua eta ebaluazioa

Eredua entrenatzeko *Scikit-learn* paketeko *LogisticRegression* funtzioa erabili dugu [31]. Funtzio honek hainbat parametro erabiltzen ditu, hala nola, 'C' erregularizazioa kontrolatzeko, 'solver' optimizazio-algoritmoa aukeratzeko, 'max_iter' iterazioen gehienezko kopurua zehazteko, besteak beste. Parametro hauek malgutasun gehigarria eskaintzen dute ereduaren errendimendua eta iragarpenak hobetzeko.

Hiru parametro horiek aldatuz, eredu desberdinak sortu ditugu eta balidazio datu multzoaren bidez, ebaluatu egin ditugu. Emaitza hoberenak 'C = 1' eta 'max_iter = 100' hautatzean lortu ditugu, defektuzkoak, ez dugulako hobekuntza handirik nabaritu hauek txikiagotu edo handitu ahala. Optimizazio-algoritmoa 'solver = liblinear' aukeratu dugu egokia baita sailkapen bitarretarako [32].

Hauek dira lorturiko emaitzak:




**4.1. Irudia**: Erregresio logistiko ereduaren emaitzak.

| ALGORITMOA | NAIVE BAYES | LOGISTIKOA |
|---|---|---|
| **PR_AUC** | 0.52 | **0.59** |
| ROC_AUC | 0.92 | 0.94 |
| Zehaztasuna | 0.92 | 0.95 |
| Prezisioa | 0.41 | 0.68 |
| Sentikortasuna | 0.73 | 0.40 |
| Espezifikotasuna | 0.93 | 0.99 |

**4.1. Taula**: Ereduen ebaluazio metriken laburpena.

Doitutako erregresio logistiko ereduak 0.95-ko zehaztasuna du. Aipagarria da espezifikotasunaren balioa ia bikaina dela, 0.99-koa. Oraingoan, sentsibilitateak 0.40 da. Aldiz, prezisioa hobetu egin da 0.68-ko balioa hartuz. ROC kurbaren AUC-a balio altua dauka, 0.94-koa. Precision-Recall kurbak 0.59 balioko AUC-a du.

Laburbilduz, erregresio logistiko ereduak Naive Bayesek baino emaitza onuragarriagoak adierazten ditu. Naive Bayes eredua, lanean zehar landutako beste ereduak ez bezala, generatiboa da. Hau da, klase bakoitzarentzat klasea hoberen deskribatuko duen eredu probabilistikoa lortzea du helburu, ondoren klasifikazio problema ebazteko. Beste ereduak, berriz, diskriminatiboak dira; helburutzat klaseak hoberen bananduko dituen eredu probabilistikoa lortzea dute. Horrek adierazten du erregresio logistikoa aproposagoa dela sailkapen-zeregin honetarako.

Hala ere, klaseen arteko desoreka nabarmena denez, sentikortasun eta prezisioaren balio baxuek mezu toxikoen detekzioa oraindik hobea izan daitekeela ondoriozta dezakegu.

# 5. Kapitulua

# Support Vector Machine

Euskarri bektoreko makinak edo Support Vector Machine (SVM) ikaskuntza estatistiko gainbegiratuko algoritmo multzo bat dira, Vladimir Vapnik-ek 90ko hamarkadan garatutakoak [33]. Nahiz eta hasiera batean sailkapen bitarreko metodo gisa garatu zen, haren aplikazioa sailkapen anizkoitzeko eta erregresio problemetara zabaldu egin da.

SVM-a dimentsio anitzeko espazio batean hiperplanoak egitean oinarritzen da, klaseak ahalik eta hobekien bereizteko. Hau da, algoritmo honek klaseetatik distantzia maximoa (hots, marjina) duen hiperplanoa bilatzen du. Horrela, klase bati dagozkion puntuak hiperplanoaren alde batean egongo dira, eta besteak, aldiz, hiperplanoaren beste aldean.

Atal honetan azaltzen den algoritmoaren garapen matematiko osoa ulertzeko beharrezkoa den teoria A eranskinean gehitu da.

## 5.1 Oinarri teorikoa: Klasifikazio bitarra

Izan bitez $X \subset \mathbb{R}^n$ eta $f : X \to Y = \{-1, +1\}$ funtzioa, definitu dezagun $S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset (X \times Y)^n$ multzo banagarria. Helburua multzo hori elkarrengandik bereizten duen $\mathcal{H}$ hiperplano bat aurkitzea da.

**5.1.1. definizioa.** n dimentsioko espazio batean, hiperplanoa n - 1 dimentsioko azpiespazio afina da.

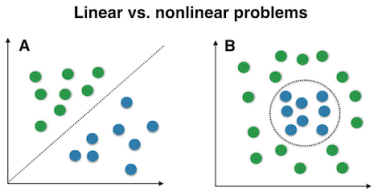
Adibidez, dimentsio bateko espazio batean hiperplano bat 0 dimentsioko azpiespazioa izango da, hau da, puntu bat. Bi dimentsioko espazioan, hiperplanoa 1 dimentsioko azpiespazioa izango da, hau da, zuzen bat. Eta hiru dimentsiokoa planoa izango da. Dimentsioak n $>$ 3 direnean, bisualizatzea zaila den arren, berdin jarraitzen du. Hemendik aurrera, hobeto ulertzeko, bi dimentsioko espazio batean lan egingo dugu, baina

kontzeptu berak aplika dakizkioke dimentsio handiagoei.

Kasu perfektu batean datuak linealki bereiz daitezke bi kategoriatan, baina hori ez da praktikoa izaten. Izan ere, bizitza errealean datuen nahasketa dela eta zaila izan ohi da. Beraz, SVMak hiru kasutarako aztertuko ditugu: linealki bananduak, linealki kuasi-bananduak eta linealki banandu ezin direnak [34][35].

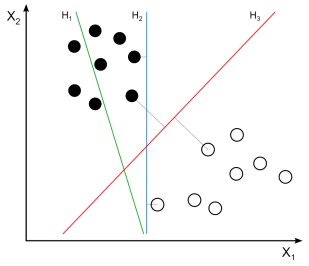
### 5.1.1 Linealki banandutako kasuak

Lehen atal honetan, linealki banandutako kasuak aztertzen dira:



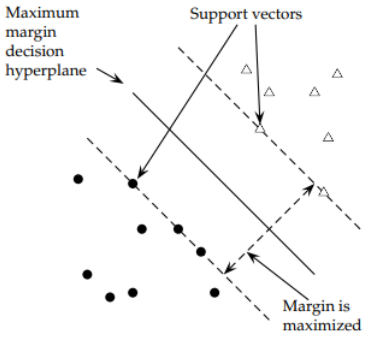

**5.1. Irudia**: A-k bi klaseak linealki banadu daitekeen problema lineala adierazten du eta B-k, berriz, linealki banandu ezin den problema.

Arestian aipatu den bezala, SVM metodoak klaseak bereizten dituen hiperplanoa aurkitzen datza. Arazoa infinitu hiperplano desberdin daudela da, eta, beraz, bi puntu motak hobekien bereizten dituena zein den jakitea da kontua. 5.2. irudiak hiru hiperplano desberdin erakusten ditu. Argi ikusten da $H_1$-ak ez dituela ezta klaseak ondo bereizten eta $H_2$-ak klaseak ondo bereizten dituen arren, puntuen eta hiperplanoaren arteko distantzia ez da egokia. Aldiz, $H_3$ hiperplano egokiena da, klaseak ondo bereizteaz gain, behar bezalako marjina uzten duelako.




**5.2. Irudia**: Hiru hiperplano ezberdin bereizten dira: $H_1$ (berdea), $H_2$ (urdina) eta $H_3$ (gorria).

Ondorioz, helburua hiperplanoa eta bi klaseen arteko marjina maximizatzen duen hiperplanoa aurkitzea da, hiperplano horri *marjina maximoko hiperplanoa* (*maximal margin hyperplane*) izena ematen zaio. Halaber, *euskarri bektore* (*support vectors*) deitzen zaizkie marjina maximoa denean sortzen diren bektoreei, ikusi 5.3. irudia.




**5.3. Irudia**: Marjina maximoko hiperplanoa eta haren alboetan euskarri bektoreak erakusten dira.

Orain, lor dezagun hiperplano baten ekuazioa. Izan bitez $\mathcal{H}$ hiperplanoa eta $\mathbf{w}$ $\mathcal{H}$ hiperplanoari perpendikularra den bektorea. $\mathbf{x} \in \mathcal{H}$ bada,

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x} = C \tag{5.1}$$

dugu non C konstante bat den. Berdintzaren bi aldeetan $\|\mathbf{w}\|$ biderkatuz eta $\|\mathbf{w}\| \cdot C = -b$ eran definituz:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{5.2}$$

$\mathcal{H}$ hiperplano baten ekuazioa lortzen dugu. Sailkapena hurrengo eran defini daiteke:

$$y_i = \begin{cases} +1, & \mathbf{w} \cdot \mathbf{x}_i + b \geq 0 \quad \text{bada} \\ -1, & \mathbf{w} \cdot \mathbf{x}_i + b \leq 0 \quad \text{bada} \end{cases} \tag{5.3}$$

Bestetik, euskarri bektoreak osatzen duten hiperplanoak aurki ditzakegu, hain zuzen ere: $\mathbf{w} \cdot \mathbf{x} + b = k$ eta $\mathbf{w} \cdot \mathbf{x} + b = -k$, $k \in \mathbb{R}$ izanik. Matematikoki komenigarria da $k = 1$ hartzea, horrela: $\mathbf{w} \cdot \mathbf{x} + b = 1$ eta $\mathbf{w} \cdot \mathbf{x} + b = -1$ lortzen ditugu. Hortaz,

$$y_i = \begin{cases} +1, & \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \quad \text{bada} \\ -1, & \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{bada} \end{cases} \tag{5.4}$$

hau da,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad i = 1, \ldots, n \tag{5.5}$$

Gure helburua marjina maximizatzea denez, bi bektore hartuko ditugu: $\mathbf{x}^+$, euskarri bektorearen $\mathbf{w} \cdot \mathbf{x} + b = 1$ hiperplanoari apuntzen diona, eta $\mathbf{x}^-$, berriz, $\mathbf{w} \cdot \mathbf{x} + b = -1$ hiperplanoari. Bi bektore horien arteko kenketa eginez, eta emaitza $\mathbf{w}$ bektore unitarioaren arteko biderkadura eskalarra eginez, marjinaren zabalera lortuko dugu:

$$\begin{aligned}(\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{\mathbf{x}^+ \cdot \mathbf{w} - \mathbf{x}^- \cdot \mathbf{w}}{\|\mathbf{w}\|} = \frac{\mathbf{w} \cdot \mathbf{x}^+ - \mathbf{w} \cdot \mathbf{x}^-}{\|\mathbf{w}\|} = \\ = \frac{(1-b) - (-1-b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}\end{aligned} \tag{5.6}$$

Beraz, $\frac{2}{\|\mathbf{w}\|}$ maximizatu behar dugu, edo baliokideki $\|\mathbf{w}\|$ minimizatu. Nola $\|\mathbf{w}\|$ ez da 0 puntuan diferentziagarria, $\frac{1}{2}\|\mathbf{w}\|^2$ minimizatuko dugu, haren deribatua $\mathbf{w}$ soilik baita.

Azken finean, $\frac{1}{2}\|\mathbf{w}\|^2$ eta 5.5 murrizketa kontuan hartuz, hiperplano optimoaren bilaketa $\mathbf{w}$ eta b balioak aurkitzeko problema gisa adieraz daiteke:

$$\begin{aligned}&\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad i = 1, \ldots, n\end{aligned} \tag{5.7}$$

Problema hau optimizazio problema bat da, problema primala. Zehatzago, optimizazio problema koadratikoa, optimizazio teoriarekin ebatz daitekeena. Nola helburu funtzioa eta murrizketak ganbiltasun kriterioa betetzen duten, teoria horrek ziurtatzen digu problema horrek problema duala duela. A eranskinean azaldu bezala, bere forma dualaren bidez ebatz dezakegu. Hurrengo funtzioa problemari dagokion Lagrangeren funtzioa da:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \tag{5.8}$$

non $\alpha_i$ *Lagrangeren biderkatzaile* izena hartzen duten.

Jarraian, *Karush-Kuhn-Tucke (KKT)* baldintzak kontuan hartuz:

- Lagrangearraren gradientea nulua

$$\frac{\partial \mathcal{L}(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, b, \alpha)}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

- Baldintza osagarria

$$\alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0 \quad i = 1, \ldots, n$$

- Murrizketa primalak

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad i = 1, \ldots, n$$

- Murrizketa dualak

$$\alpha_i \geq 0 \quad i = 1, \ldots, n$$

Lehenengo baldintzetatik hurrengoa lortzen dugu:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{5.9}$$

eta

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{5.10}$$

Orain, 5.8 adierazpenean 5.9 eta 5.10 ordezkatuz, problema dualaren helburu funtzioa lortzen dugu, funtzio Lagrangearra soilik $\alpha$-ren menpe geratuko da:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i y_i \mathbf{w} \cdot \mathbf{x}_i - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j\right) - \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j\right) - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$= -\frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j\right) + \sum_{i=1}^{n} \alpha_i$$

$$\mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{5.11}$$

Era honetan, gure problema duala lortzen dugu:

$$\begin{aligned} &\max_{\alpha} \mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{eta} \quad \alpha_i \geq 0 \quad i = 1, \ldots, n \end{aligned} \tag{5.12}$$

Problema horren soluzioa aurkitzea problema primalaren soluzioa aurkitzearen baliokidea da. Azken problema hau programazio koadratikoa erabiliz ebatz daiteke, eta nabarmendu behar da hasierako problema baino errazagoa dela.

Behin 5.12 problema dualaren soluzioa aurkitu dugula, $\alpha^*$, erraz lortzen dugu bilatutako hiperplanoaren ekuazioa:

$$\mathcal{H}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + b^* \tag{5.13}$$

non $b^*$ren balioa KKT-ren baldintza osagarria eta murrizketak erabiliz kalkula dezakegun. $\alpha_i > 0$ bada, orduan:

$$\mathbf{y}_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1 = 0 \tag{5.14}$$

Hau da, $\alpha_i > 0$ betetzen duten $\mathbf{x}_i$ bektoreak euskarri bektoreak izango dira. Beraz, $b^*$ askatuz:

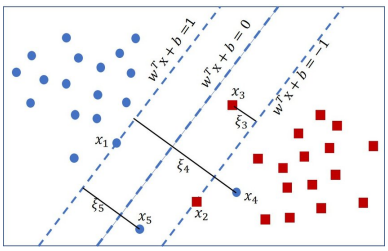
$$b^* = \mathbf{y}_{vs} - \mathbf{w}^* \cdot \mathbf{x}_{vs} \tag{5.15}$$

non $(\mathbf{x}_{vs}, \mathbf{y}_{vs})$ edozeien euskarri bektoreen puntua adierazten duen (haren klasearekin batera). Praktikan, alegia, emaitza zehatzagoa lortzeko, hobe da $b^*$-ren balioa kalkulatzea euskarri bektore guztien batezbestekotik abiatuta. Izan bedi $SV$ euskarri bektoreen multzoa eta $N_{SV}$ haren kardinala, ondorioz:

$$b^* = \frac{1}{N_{SV}} \sum_{j \in SV} (y_j - \mathbf{w}^* \cdot \mathbf{x}_j) \tag{5.16}$$

### 5.1.2 Linealki kuasi-banaduak

Orokorrean, errealitatean aurre egiten ditugun problemak, ez dira erabat linealki banagarriak. Hori dela eta, aldaketa batzuk egin behar zaizkio aurreko funtzioari. Kasu hauetan, jarraituko den prozedura laginaren datuak klasifikatzean, errore kopuru txiki bat egotea da. Hori lortzeko, murrizketetan aldagai erreal positiboak gehitu behar dira, $\xi_i \quad i = 1, \ldots, n$, *lasaiera aldagaiak* izenak dutenak. $\xi_i$ $(x_i, y_i)$ bikoteari dagokion desbideratzea neurtuko du, $y_i$ klaseari dagokion marjinatik neurtuta, ikusi 5.4. irudia.




**5.4. Irudia**: $\xi_3$, $\xi_4$ eta $\xi_5$ aldagaiek desbideratzeak erakusten dituzte.

Beraz, murrizketa honakoa izango litzateke:

$$y_i(w \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{5.17}$$

Oraingoan, gure helburua marjina maximizatzea izateaz gain, aldi berean, $\sum_{i=1}^{n} \xi_i$ minimoa izan behar da. Beraz, gure optimizazio problemaren helburu funtzioa hurrengoa da:

$$f(\mathbf{w}, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

non C *erregularizazio parametroa* den. Parametro hau eskuz aukeratzen da, $\xi_i$-ei eman nahi zaien garrantziaren arabera. C ren balio handi batek $\xi_i$-ei balio txikiak emango ditu; hori marjina estuago batean gertatuko litzateke, entrenamendu datuetara gehiegi doituta, eta overfitting arazoak sor ditzake. Aldiz, C balio handiak $\xi_i$ balio handiagoak onartzen baditu, marjinaren zabalera handiagoa izango du, eta puntu gehiago marjinaren barruan kokatuko dira, gaizki sailka daitezkeenak.

Laburbilduz, gure optimizazio problema hurrengoa da:

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ &y_i(\mathbf{w}\cdot\mathbf{x}_i + b) + \xi_i - 1 \geq 0, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{5.18}$$

Era honetan definitutako hiperplanoa *marjina leunekoa (soft margin)* dela esaten da, eta linealki banadutako kasuan, aldiz, *marjina zurruneko (hard margin)* hiperplano izena hartzen zuen. Problema honen ebazpena aurreko ataleko kasuaren antzekoa da, baina bi Lagrangeren biderkatzaile izanik. Azalpen osoa hemen aurki daiteke: [35]. Lortzen den problema duala honakoa da:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \quad \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \end{aligned} \tag{5.19}$$

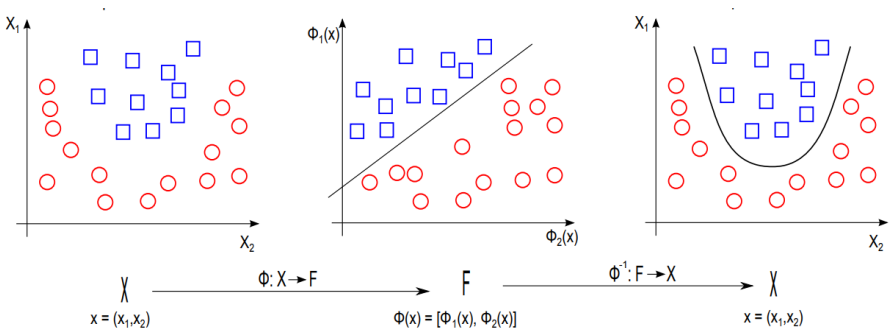
Behin problema ebatzita, eta $\alpha^*$ problema dualaren soluzio optimoa izanik, problema primalaren hiperplano optimoaren $\mathbf{w}$ eta $b$ balioak lor ditzakegu.

Amaitzeko, $0 < \alpha_i^* < C$ bada, orduan $\xi_i = 0$ banagarriak diren adibideei dagokie, marjinaren mugan kokatuta daudenak, *free support vectors* deritze. $\alpha_i^* = C$ bada, ordea, $\xi_i > 0$, banagarriak ez diren adibideei dagokie. Zuzen sailkatuta egongo dira $0 < \xi_i < 1$ eta oker sailkatuta $\xi_i > 1$ bada (*bounded support vectors*).

### 5.1.3 Linealki banangarriak ez diren kasuak

Datu multzoa ezin denean linealki bereizi, sarrera-espazioa transformatuko dugu funtzio ez-lineal baten bidez, dimentsio handiko espazio batean,

karakteristiken espazioan $\mathcal{F}$. Espazio horretan banangaria den hiperplano optimoa bilatuko dugu.




**5.5. Irudia**: Sarrera-espazioan funtzio ez-lineal bat bilatzearen problema, karakteristiken-espazioan funtzio lineal (hiperplano) bat bilatzeko problema berri batean bihur daiteke. Hiperplano hori funtzio ez-lineal batean transformatuko da jatorrizko sarrera-espazioan.

Izan bedi $\Phi : X \to \mathcal{F}$ transformazio funtzioa non $\mathbf{x} \in X$ bektore bakoitzari puntu bat esleitzen zaion $\mathcal{F}$ karakteristika espazioan, non $\Phi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \ldots, \varphi_m(\mathbf{x})]$ den eta existitzen da $\varphi_i(\mathbf{x})$ funtzio ez-lineala $i \in 1, \ldots, m$.

Testuinguru horretan, honela adierazten da karakteristiken espazioko hiperplano banatzailearen ekuazioa:

$$\mathcal{H}(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) \tag{5.20}$$

Ohartu $b$ parametroa baztertu dezakegula, izan ere, transformazio funtzioan sar daiteke funtzio konstante bezala, $\varphi_1(\mathbf{x}) = 1$. Beraz, hurrengoa dugu:

$$\mathcal{H}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle \tag{5.21}$$

**5.1.2. definizioa.** *Kernel funtzioa* $K : X \times X \to \mathbb{R}$ funtzio bat da non $X$ sarrera espazioko bikote bakoitzari balio erreal bat esleitzen zaion, $\mathcal{F}$ karakteristiken espazioan. Balio erreal hori, elementuen irudien arteko biderkadura eskalarra eginez lortzen da. Hau da,

$$K(\mathbf{x}_1, x_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle = \Phi_1(\mathbf{x}_1)\Phi_1(\mathbf{x}_2) + \cdots + \Phi_m(\mathbf{x}_1)\Phi_m(\mathbf{x}_2) \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in X$$

non $\Phi : X \to \mathcal{F}$ dagokion transformazio funtzioa den.

Ondorioz, biderkadura eskalarra ordez, Kernel funtzioa jar dezakegu:

$$\mathcal{H}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) \tag{5.22}$$

Ebazteko erabiliko dugun problema duala aurreko ataleko berdina da, baina biderkatura eskalarra Kernel funtzioarekin ordezkatuz:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \end{aligned} \tag{5.23}$$

non $K(\mathbf{x}_i, \mathbf{x}_j)$ Kernelen funtzioa den.

Hurrengo teoremak erantzuna ematen digu dimentsio finituko sarrera-espazioa dimentsio infinituko espazio bihurtzeko:

**5.1.1. teorema. Moore Aronszajn-ren teorema**
Izan bedi $K : X \times X \to \mathbb{R}$ funtzio simetrikoa eta positiboki erdidefinitua. Orduan, $K$ funtziorako existitzen da Hilbert espazioa eta $\Phi : X \to \mathcal{F}$ funtzioa non $K(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$ den $\forall \mathbf{x}_1, \mathbf{x}_2 \in X$.

Teorema honetatik abiatuta, ondoriozta dezakegu kernel funtzio bat eraikitzeko ez dela beharrezkoa $\Phi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \ldots, \varphi_m(\mathbf{x})]$ oinarrizko funtzio multzo batetik abiatuta eraikitzea. Azken finean, 5.23 problema duala ebazteko, ez da beharrezkoa oinarrizko funtzioen multzo hori ezagutzea, ezta ezaugarrien espazioan transformatutako koordenatuak ezagutzea ere. Nahikoa da dagokion Kernel forma jakitearekin, honi *kernel trikimailu (Kernel trick)* izena ematen zaio.

Hona hemen kernel funtzioen adibide ohikoenak:

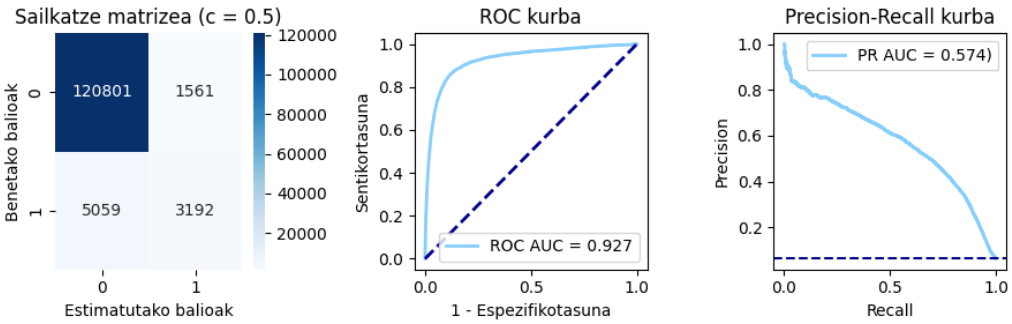
- Kernel lineala:
$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}' \tag{5.24}$$
- Kernel polinomikoa (p - gradukoa):
$$K(\mathbf{x}, \mathbf{x}') = [\gamma(\mathbf{x} \cdot \mathbf{x}') + \tau]^p, \quad \gamma > 0 \tag{5.25}$$
- Kernel gaussiarra:
$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2), \quad \gamma > 0 \tag{5.26}$$

  non $\gamma$, $\tau$ y $p$ parametroak Kernelen parametroak diren.

## 5.2 Entrenamendua eta emaitzak

SVM eredua inplementatzeko *LinearSVC* paketea erabili dugu, hau ere *Skicit-learn*-ekoa [36]. Pakete horrek klasifikazio linealeko problemetarako SVMen inplementazio efizientea eskaintzen du, eta bereziki erabilgarria da datu-multzo handiekin lan egiten denean. *Skicit-learn*-eko *SVC* paketeak kernela zehazteko aukera ematen du, baina datu handiekin ez digu funtzionatu.

Eredua doitzean, aldatu dugun parametro bakarra 'max_iter' izan da, 1000-tik 4000-ra handitu dugu, entrenamenduan iterazio gehiago egiteak, algoritmoaren konbergentzia egokia bermatzen duelako. Eredua entrenatzeko behar den denbora handitu den arrren, ondo doitutako eta gure datuen sailkapenean errendimendu ona duen SVM eredu bat lortu dugu, ondoren azaltzen den bezala:




**5.6. Irudia**: Support Vector Machine ereduaren emaitzak.

| ALGORITMOA | NAIVE BAYES | LOGISTIKOA | SMV |
|---|---|---|---|
| **PR_AUC** | 0.52 | **0.59** | 0.57 |
| ROC_AUC | 0.92 | 0.94 | 0.93 |
| Zehaztasuna | 0.92 | 0.95 | 0.95 |
| Prezisioa | 0.41 | 0.68 | 0.67 |
| Sentikortasuna | 0.73 | 0.40 | 0.39 |
| Espezifikotasuna | 0.93 | 0.99 | 0.99 |

**5.1. Taula**: Ereduen ebaluazio metriken laburpena.

Bistakoa da emaitza hauek eta erregresio logistikoko ereduarekin lortutakoak oso antzekoak direla. Hau da, zehaztasun balio altua, espezifikotasun balio ia ezin hobea, prezisioa nahikoa eta sentikortasun balio okerra. ROC eta Precision-Recall kurbetan adierazten diren AUC-ak ere gutxi aldatu dute, balio pixka bat baxuagoak hartuz, 0.93 eta 0.57, hurrenez hurren. Beraz, berriz ondoriozta dezakegu, nahiz eta ereduak 0 klasean oso ondo iragaztzen dituen, eredua hobe daitekeela 1 klasea aurreikusteko.

# 6. Kapitulua

# Decision Tree eta Random Forest

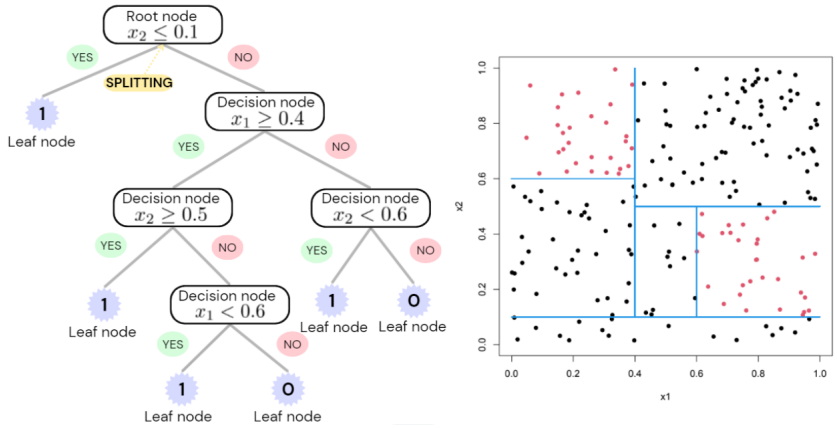
Erabaki zuhaitzak (Decision Tree) ikaskuntza gainbegiratuko algoritmoak dira, klasifikazio eta erregresio lanak egin ditzaketenak. Erabaki zuhaitza laginaren espazioa zatitzean oinarritzen den metodo hierarkikoa da, zuhaitz egitura duen diagraman irudikatuz. Ideia nagusia erantzun aldagaia gehien banatzen duen aldagaia aurkitu eta aldagai horren arabera lagina zatitzean datza, ondoren laginaren zati bakoitzean prozesua errepikatuz. Indibiduo berri baten klasea aurreikusteko, zuhaitzaren banaketa jarraitu besterik ez dugu egin behar [37][38].

Erabaki zuhaitzak algoritmo indartsuak dira, konputazionalki interpretatzeko errazak eta eraginkorrak direlako. Hala ere, orokorrean, emaitza hobeak lortzen dituzten algoritmoak ausazko basoak (Random Forest) dira, erabaki zuhaitzen multzoa baitira [39][40].

## 6.1 Oinarri teorikoa

Hasteko, ezinbestekoa da erabaki zuhaitzak ulertzea. Aurretik aipatu bezala, algoritmo honek erantzun aldagaia estimatzeko lagina zatitzen du, zuhaitz batetako adarretan sakabanatuz.

Zuhaitz hauek hainbat nodoz osatuta daude: *erro-nodoa (root node)* lagin osoa irudikatzen du, *erabaki-nodoa (decision node)* nodo bat banatzean sortutako azpinodoa da, eta bukatzeko, *hosto-nodoa (leaf node)*. Barne-nodo bakoitzak baldintza bat irudikatzen du eta hosto bakoitzak, aldiz, dagokion etiketa. Nodo bat bi azpinodotan edo gehiagotan banatzeko prozesuari *zatiketa (splitting)* deritzo. 6.1. irudia zuhaitz simple baten adibidea litzateke.




**6.1. Irudia**: Erro-nodoak $x_2 \leq 0.1$ baldintza adierazten du eta baldintza hori betetzekotan, lagina 1 (beltza) klasekoa dela iragartzen da. Bestela, beste baldintza bat ezartzen da. Horrela, lagina zatitzen jarraitzen da, datu guztiak klase berdinekoak izan arte. Hau da, 1 (beltza) edo 0 (gorria) klasera heldu arte. Grafikoari erreparatuz, laginak erabaki zuhaitzean adierazitako zatiketekin bat datozela ikusten da.

Jarraian, erabaki zuhaitzak eratzeko algoritmoa zertan oinarritzen den aztertzen da. $i(t)$ *ezpurutasun-neurri* batek $t$ nodo bakoitzaren kalitatea ebaluatzen du. Hau da, nodoan klaseak zenbat nahasten diren neurtzen du. Helburua nodoak ahalik eta puruen izatea da, homogeneoak, alegia. Beraz, $i(t)$ ezpurutasun-neurria txikitzea bilatuko da, zatiketa zehatzagoak lortzeko.

Honela definitzen da $t$ nodoa ezkerreko nodo ($t_L$) eta eskuineko nodo ($t_R$) batean banatzen duen $s \in Q$ zatiketa bitarraren ezpurutasunaren murrizketa:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{6.1}$$

non $p_L$ $\frac{N_{t_L}}{N_t}$-ren proportzioa den, $N_{t_L}$ ezkerreko nodora doazen laginen kopurua izanik eta $N_t$ $L_t$ datu multzoak duen tamaina (hurrenez hurren, $p_R$).

Hori kontuan hartuz, horrela adierazten da $s^*$ zatiketa optimoa:

$$s^* = \arg\max_{s \in Q} \Delta i(s,t) \tag{6.2}$$

Izan ere, $s^*$-k zatitutako nodoetan purutasuna edo homogeneotasuna ahalik eta gehien handitzea du helburu.

Klasifikazio zuhaitzetarako, ezpurutasun-irizpide ohikoenak *Shannon-*

*en entropia* [41] eta *Gini indizea* [42] dira, hurrenez hurren:

$$i_H(t) = -\sum_{k=1}^{J} p(c_k|t) \log_2(p(c_k|t)) \tag{6.3}$$

$$i_G(t) = \sum_{k=1}^{J} p(c_k|t) \left(1 - p(c_k|t)\right) \tag{6.4}$$

non $p(c_k|t)$ $t$ nodoa $c_k$ klasekoa izateko probabilitatea den eta nodoan dauden $c_k$ klaseko lagin proportzioarekin hurbiltzen den.

Erabaki zuhaitz baten eraikuntza geldiarazteko unea erabaki behar da baita ere. Prozesu hori, oro har, geldialdi irizpideetan oinarritzen da, hala nola: nodo bateko lagin guztiak homogeneoak direnean, zuhaitzaren gehieneko sakonera finkatuta duenean edo ezpurutasunaren murrizketa totala aukeratutako atalase finko bat baino txikiagoa denean, besteak beste.

Behin $t$ nodoa hosto-nodoa dela erabakitzean, $\hat{y}_t$ balio konstante bat esleitzen zaio, gehiengoaren klasearen arabera:

$$\hat{y}_t = \arg\max_{c \in Y} p(c \mid t) \tag{6.5}$$

Ondorioz, zuhaitza eraikitzeko prozesua algoritmo iteratibo baten bidez bistara daiteke, aipatutako kontzeptu guztiak laburbiltzen dituena:

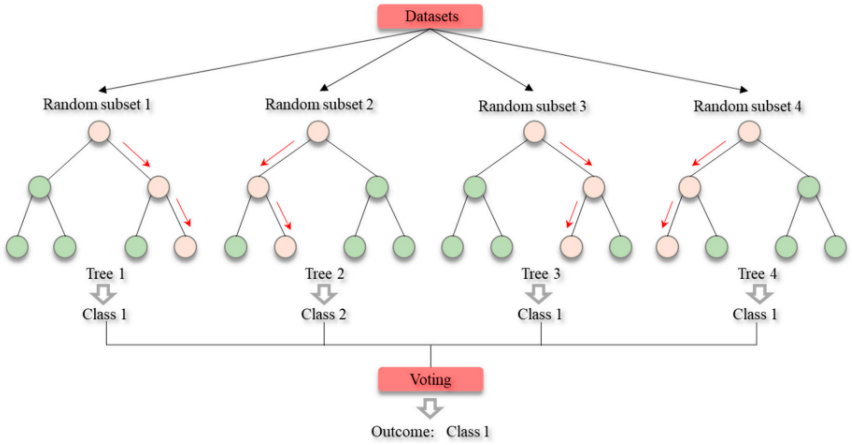
**Algoritmoa 1 : Erabaki zuhaitza**

**function** DecisionTree($L$)
  Sortu $\phi$ erabaki zuhaitza $t_0$ erro nodoarekin.
  $(t, L_t)$ nodo irekien $S$ pila huts bat sortu.
  $S$.push($(t_0, L)$)
  **while** $S$ ez bada hutsa **do**
    $(t, L_t) \leftarrow S$.pop()
    **if** geldialdi-irizpidea betetzen da $t$-rentzako **then**
      $\hat{y}_t$ (balio konstantea)
    **else**
      $L_t$ ren zatiketa bilatu, ezpurutasun gehien murrizten duena.
      $s^* = \arg\max_{s \in Q} \Delta i(s, t)$
      $L_t$ zatitu $L_{tL} \cup L_{tR}$ nodoetan $s^*$-ren arabera
      Sortu $t$-ren ezker nodoa $t_L$
      Sortu $t$-ren eskuin nodoa $t_R$
      $S$.push($(t_R, L_{tR})$)
      $S$.push($(t_L, L_{tL})$)
    **end if**
  **end while**
  **return** $\phi$
**end function**

Erabaki zuhaitzek desabantaila handi bat dute: gehiegizko doikuntza (overfitting). Hau da, oro har, eraginkortasunez egokitzen dira entrenamendu

datuetara, baina datu berriak sartzerako orduan, aurresanean akats ugari egiten dituzte.

Muga hori gainditzeko, ausazko basoak sortu ziren, multzo gisa lan egiten duten erabaki zuhaitz ugariz osotuak. Zuhaitzak eraikitzeko zoriz aukeratutako datuen azpimultzoa erabiltzen da eta horrela, overfitting-a konpontzen da, zuhaitz bakoitzean aldagaiak desberdinak izango direlako. Algoritmoaren iragarpena banakako zuhaitzen klase-iragarpenen artean boto gehien duena izango da, ikusi 6.2. irudiko adibidea.




**6.2. Irudia**: Random Forest adibidea: ausaz hautatutako lau erabaki zuhaitz sortzen dira eta zuhaitz bakoitzaren adarrak jarraituz iragarritako klasea ondorioztatzen da, esate baterako, lehenengoko zuhaitzean 1 klasea. Amaitzeko, 1 klaseak hiru boto dituenez eta 2 klaseak, aldiz, bakarra, 1 klasekoa dela ondorioztatzen da.

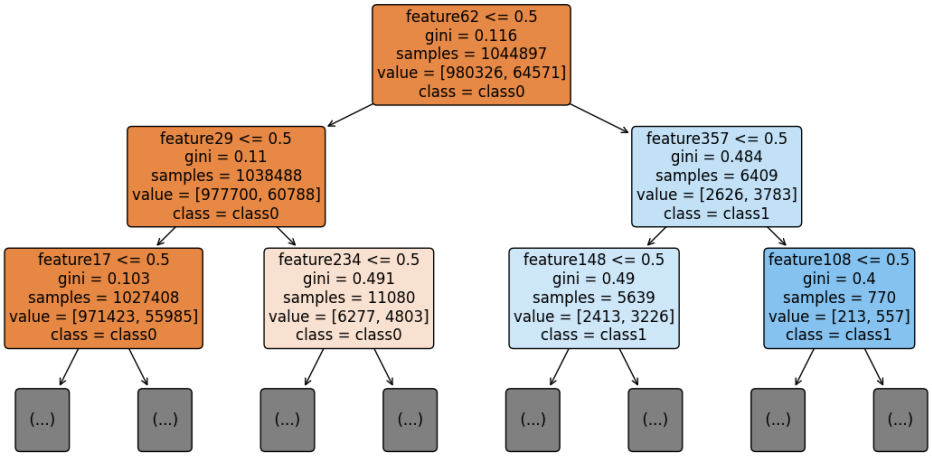
## 6.2 Entrenamendua eta ebaluazioa

Ereduaren entrenamendurako *scikit-learn* liburutegiko *DecisionTreeClassifier* [43] eta *RandomForestClassifier* [44] moduloak erabili dira.

Bi klasifikatzaileek zenbait parametro partekatzen dituzte, adibidez, 'criterion', zatiketa baten kalitatea neurtzeko funtzioa, 'max_depth', zuhaitzaren gehieneko sakonera eta 'min_samples_split', barne-nodo bat zatitzeko behar den gutxieneko lagin kopurua.

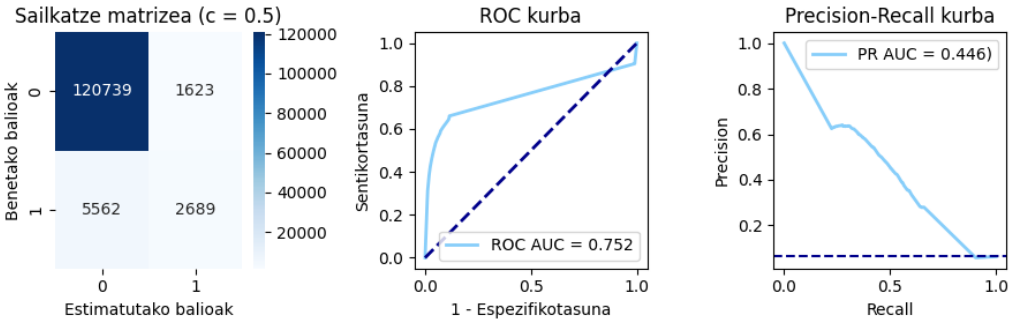
Entrenamenduan *gini* eta *entropy* irizpideak frogatu ondoren, bien arteko aldea oso txikia dela ikusi da balidazio multzoan. Bestalde, exekuzioak denbora luzea behar zuenez, eta emaitzak oso onak ez zirenez, zuhaitzaren

gehiegizko sakonera ezartzea erabaki da. Horri esker, entrenamenduaren exekuzio denbora nabarmen murriztu da; eta emaitzak apur bat hobetu dira. Azkenik, *gini* eta 'max_depth = 70' hautatu ditugu. 6.3. erabaki zuhaitz ereduaren lehenengo nodoak ikus daitezke.




**6.3. Irudia**: Giniren indizea 0 (lagin guztiak klase berdinekoak) eta 0.5 (klase-nahaste perfektua) bitartekoa da nodo batean. 'Samples' balioak nodo espezifiko horretan zenbat datu-lagin dauden zehazten du eta 'value'-k nodoaren barruan mota bakoitzean dagoen lagin-kantitatea. 'Class' nodo horretako klase nagusia da. Adibidez, erro-nodoan, Giniren indizea 0.116 da; guztira, 1044897 lagin daude, 980326 lagin 0 klasekoak (ez toxikoak) dira eta 64571 lagin 1 klasekoak (toxikoak), 0 klasea nagusia delarik.

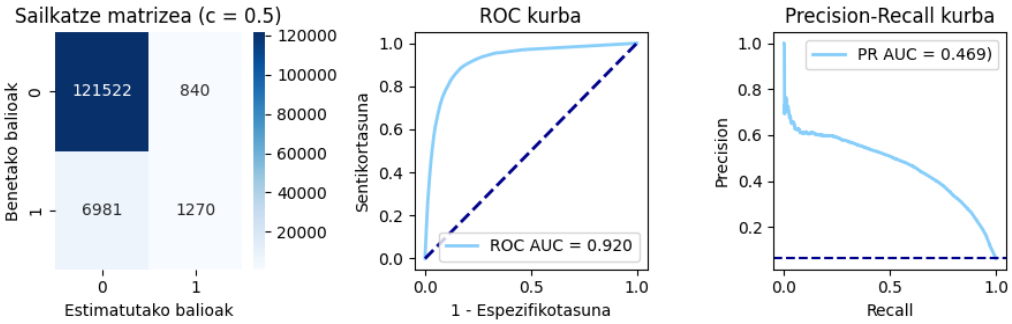
6.4. irudiak Decision Tree ereduak test-eko datuetan lortutako emaitzak erakusten ditu.




**6.4. Irudia**: Erabaki zuhaitza (Decision Tree) ereduaren emaitzak.

6.4. irudiak erakusten digu ROC kurbaren AUC-a esanguratsuki baxua dela, 0,76-koa, momentuz lortu ditugunetik txarrena. Hori, bi klaseen aurresateko gaitasuna nahiko hobe daietekeela adierazten digu. Era berean, Precision-Recall kurbaren AUC-a eskasa da, 0,45-koa, hau da, errendimendu eskasa klase positiboaren sailkapenean.

Ausazko basoaren eredua entrenatzeko, arazo larri bat izan dugu exekuzio denborarekin eta laginen tamaina murriztu behar izan dugu, milioi bat mezu inguru izatetik, 256000-ko lagina izatera. Entrenamendu datuak laurden batetara murriztu arren, eredu honekin 6.5. irudian erakusten diren emaitzak lortu dira.




**6.5. Irudia**: Ausazko basoen (Random Forest) ereduaren emaitzak.

Ereduaren emaitzak ez dira onenak, laginaren laurdena bakarrik kontsideratu dugula kontuan hartuz, seguruenik ereduaren sailkapenean eragina izan dezakena. Hala ere, ROC kurbaren azpiko azalera nabarmen hobetzen da erabaki zuhaitzarekin alderatuta. Horrez gain, Precision-Recall kurbaren AUC-ak adierazten du klase positiboaren iragarpen hobea egiteko balio handiagoa lor daitekeela. Oro har, teorian azaldu bezala, ausazko basoek erabaki zuhaitzak baino emaitza hobeak lortzen dituzte, hainbat zuhaitz konbinatzen baititu.

Hurrengo taulan emaitz guztiak biltzen dira:

| ALGORITMOA | NAIVE BAYES | LOGISTIKOA | SMV | DECISION TREE | RANDOM FOREST |
|---|---|---|---|---|---|
| **PR_AUC** | 0.52 | **0.59** | 0.57 | 0.45 | 0.47 |
| ROC_AUC | 0.92 | 0.94 | 0.93 | 0.75 | 0.92 |
| Zehaztasuna | 0.92 | 0.95 | 0.95 | 0.94 | 0.94 |
| Prezisioa | 0.41 | 0.68 | 0.67 | 0.62 | 0.60 |
| Sentikortasuna | 0.73 | 0.40 | 0.39 | 0.33 | 0.15 |
| Espezifikotasuna | 0.93 | 0.99 | 0.99 | 0.99 | 0.99 |

**6.1. Taula**: Ereduen ebaluazio metriken laburpena.

# 7. Kapitulua

# Ondorioak

Aurreko kapituluetan jasotako informazio guztia aztertu eta hausnartu ondoren, hainbat ondorio atera ditugu, hurrengo lerroetan azaltzen direnak.

Hasteko, esan beharra dago, Machine Learning-eko klasifikazio algoritmoak tresna indartsuak direla. Algoritmo bakoitzak bere abantailak eta desabantailak ditu, eta eraginkortasuna aldatu egiten da problema espezifikoaren eta eskura dauden datuen arabera. Algoritmo egokia hautatzea hainbat faktoreren menpe dago, algoritmoak datuen berezitasunen arabera egokitzen baitira.

Lan honetan, bost algoritmo ezberdin erabili ditugu gure datuak mezu toxikoak diren edo ez detektatzeko: Naive Bayes, erregresio logistikoa, Support Vector Machine (SVM), Decision Tree eta Random Forest.

Jarraian, algoritmoek jasan ditzaketen arazoak azaltzen dira. Klasifikazio algoritmoak sendoak diren arren, hainbat arazo esanguratsuri egin behar diete aurre, hala nola: prozesatze-denbora luzea izan ditzakete, Random Forest-aren kasuan bezala; datu handiek kostu konputazional handiak eragin ditzakete, memoria arazoak bektorizatze prozesuan gertatzen zen moduan; overfitting-a jasan dezakete, Decision Tree bezalako algoritmoek; datuak aurreprozesatzeko etapan, berriz, informazio galera handia egon daiteke, puntuazio markek, adibidez, esanahia guztiz alda dezakete. Gainera, algoritmo batzuek, esate baterako, Naive Bayesek, aldagaien independentzia supositzen dute, eta hori ez da, oro har, egia. Funtsezkoa da erronka horiek kontuan izatea, sailkapen-ereduen errendimendua optimizatzeko.

Lortutako emaitzei erreparatuz, erregresio logistikoa eta SVM izan dira eredurik eraginkorrenak, emaitza hoberenak lortu dituztenak. Ondoren, Naive Bayes eredua eta azkenik, Random Forest eta Decision Tree. Orokorrean, algoritmo guztiek espezifikotasun ia bikaina dute, hau da,

ereduak 0 klasea (ez-toxikoak) oso ondo sailkatzen du. Eredu guztietan prezisioa eta sentikortasuna dauzkate balio baxuenak, honek 1 klasea (toxikoa) eskas iragartzen duela adierazten digu. Zehaztasuna oso ona da, hots, ereduak ondo sailkatzen du mezuen proportzio handi bat, gure datuak desorekatuak baitaude eta 0 klasea da nagusi. Hau AUC estatistikoan islatu egiten da. ROC AUC altua den arren, normalean ereduaren aurresateko gaitasun ona adierazten duena, Precision-Recall kurbak erakusten du ereduak ez duela guztiz ondo sailkatzen klase minoritarioa (toxikoak).

Ondorioz, klaseak desorekatuak dituzten problemetan, funtsezkoa da sentikortasuna eta prezisioa bezalako metriketan zentratzea, hobeto islatzen baitute ereduaren errendimendua gutxienengo klasean. Gure kasuan, ondoriozta dezakegu ereduek Precision-Recall kurbaren AUC-a nahiko hobetu dezaketela, agian, klaseen arteko orekatze teknikak edota beste algoritmo egokiagoak erabiliz.

Bukatzeko, azpimarratu nahiko nuke, algoritmoen ulermen sakonak matematikaren ezagutza sendoa eskatzen duela. Batez ere, probabilitatea funtsezkoa da ereduek iragarpenak nola erabiltzen dituzten ulertzeko. Aljebra lineala beharrezkoa da algoritmo askoren oinarrian dauden eragiketa matrizialak eta transformazioak ulertzeko. Estatistikak, berriz, ereduaren errendimendua interpretatzeko eta ebaluatzeko oinarriak ematen ditu. Graduan zehar ikasitakoari esker, oinarri teoriko sendoa duten ereduak garatzeko eta eraginkortasunez lantzeko gai gara.

Laburbilduz, klasifikazio algoritmoek potentzial handia dute, dena dela, ezinbestekoa da hauen mugak ezagutzea, matematikoki ulertzea eta egoki ebaluatzea, eredurik eraginkorrena lortzeko.

# A. Eranskina

# Optimizazio teoria

Eranskin honetan, optimizazio teoriaren ideia nagusien laburpena egiten da, SVM-rekin lotutako problemak ebazteko.[45]

Izan bitez $f$ eta $g_i$, $i = 1, \ldots, n$ funtzio ganbilak, hurrengo optimizazio problemari **problema primal** deritzogu:

$$\begin{aligned} &\min f(\mathbf{x}), \quad \mathbf{x} \in \Omega \\ &\text{s.a.} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n \end{aligned} \tag{A.1}$$

non $f$ optimizatu beharreko funtzioa (helburu funtzioa) den eta $g_i$ murrizketak. Problema primalaren soluzioa, $\mathbf{x}^*$, $g_i(\mathbf{x}^*) \leq 0$ y $f(x^*) \leq f(x)$ $\forall x$-rako beteko du $g_i(x) \leq 0$ izanik, $i = 1, \ldots, n$.

**Lagrangeren funtzioa** honela definitzen da:

$$\mathcal{L}(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{x}) \tag{A.2}$$

non $\alpha_i$ koefizienteak **Lagrangeren biderkatzaileak** diren. Funtzio horrek optimizatu beharreko funtzioa eta murrizketa funtzioak funtzio bakar batean biltzen ditu. Lagrangeren funtzioa erabiliz, honela defini daiteke **problema duala**:

$$\begin{aligned} &\max_{\alpha} \varphi(\alpha) = \inf_{\mathbf{x} \in \Omega} \mathcal{L}(\mathbf{x}, \alpha) \\ &\text{s.a.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{A.3}$$

Optimotasunaren teoriak bermatzen du, zenbait baldintzatan, problema duala ebaztea hari lotutako primalaren ebazpena aurkitzearen baliokidea dela, eta alderantziz. Transformazio horren abantaila da, normalean, problema duala errazagoa dela ebazten primala baino. Hurrengo bi teoremek bi problemen ebazpenen arteko erlazioa erakusten dute.

**A.0.1. teorema.** Izan bitez $\mathbf{x}$ eta $\alpha$ bi bektore problema primalaren eta dualaren murrizketak betetzen dituztela, hau da, $g_i(\mathbf{x}) \leq 0$ eta $\alpha_i \geq 0$ $i = 1, \ldots, n$, orduan $\varphi(\alpha) \leq f(\mathbf{x})$.

Aurreko teorematik bi korolario ondoriozta daitezke.

**A.0.2. korolarioa.** Problema primalak goitik bornatzen du problema duala.

**A.0.3. korolarioa.** $\varphi(\alpha) = f(\mathbf{x})$ berdintza betetzen bada, orduan $\mathbf{x}$ eta $\alpha$ problema primalaren eta dualaren ebazpenak dira, hurrenez hurren.

Teorema hori interesgarria da problema primalaren eta problema dualaren arteko erlazio bat definitzeko eta bien ebazpena baliokidea izateko. Beraz, soluziotik hurbilago egongo gara, diferentzia txikiagoa den heinean, hau da, emaitza lortzen da $|\varphi(\alpha) - f(\mathbf{x})|$ nulua denean. Soluzio hori funtzio lagrangearraren zeladura-puntu bati dagokio, aldi berean $\mathcal{L}(\mathbf{x}, \alpha)$-ren minimoa $\mathbf{x}$-rekiko, eta $\mathcal{L}(\mathbf{x}, \alpha)$-ren maximoa $\alpha$-rekiko izateagatik karakterizatzen da.

Bigarren teoremak, Karush-Kuhn-Tucker-en teorema deritzonak, $\mathbf{x}^*$ problema primalaren emaitza izateko behar diren baldintzak ezartzen ditu.

**A.0.4. teorema. Karush-Kuhn-Tucker-en teorema**
Problema primalean, A.1, $f : \mathbb{R}^d \to \mathbb{R}$ eta $g_i : \mathbb{R}^d \to \mathbb{R}$ $i = 1, \ldots, n$ funtzioak ganbilak badira eta $\alpha_i \geq 0$ $i = 0, \ldots, n$ konstanteak existitzen badira non:

$$\begin{aligned} \frac{\partial f(\mathbf{x}^*)}{\partial x_j} + \sum_{i=1}^{n} \alpha_i \frac{\partial g_i(\mathbf{x}^*)}{\partial x_j} = 0, \quad j = 1, \ldots, d, \\ \alpha_i^* g_i(\mathbf{x}^*) = 0, \quad i = 1, \ldots, n, \end{aligned} \tag{A.4}$$

orduan, $(\mathbf{x}^*, \alpha^*)$ problemaren soluzio optimoa da.

Lehenengo baldintza, $\varphi(\alpha)$-ren definiziotik lortzen da, hau da, lagrangearraren funtzioaren infimo gisa definitzearen ondorioz, non x-rekiko deribatu partzialek zero izan behar baitute. Bigarrena, aldiz, baldintza osagarri deritzonak, problema primalaren eta dualaren optimoak bat datozela bermatzen du $(\varphi(\alpha^*) = f(x^*))$, izan ere, baldintza hori egia bada, funtzio lagrangearraren batukariaren batugai guztiak nuluak izango lirateke.

Karush-Kuhn-Tucker-en teoremaren interesa zera da: problema duala erabiliz, primala ebazteko jarraitu behar diren urratsak ezartzen dituela. Horrela, problema primaletik abiatuta, funtzio lagrangearra eraikitzen da. Ondoren, KKT-ren teoremaren lehen baldintza aplikatzen zaio funtzio horri,

eta, horri esker, erlazio-multzo bat lor daiteke. Erlazio horiek, Lagrangeren funtzioan ordezkatuta, funtzio horren aldagai asko desagerraraziko dituzte. Pausu hori $\inf_{\mathbf{x}\in\Omega}\mathcal{L}(\mathbf{x},\alpha)$ kalkulatzearen baliokidea da. Horrela lortutako funtzio duala Lagrangeren biderkatzaileen araberakoa izango da soilik. Baliteke, halaber, lehen KKT baldintza aplikatzean lortutako erlazio-multzotik, murrizketa gehigarriak sortzen dira aldagai dualentzat.

Azkenik, problema primalaren emaitza lortzeko, problema duala ebaztean lortutako emaitza ordezkatuz lortzen dugu, lehen KKT baldintza funtzio lagraniarrari aplikatzean lortutako erlazioetan.

$f$ funtzioa koadratikoa bada eta $g_i$ murrizketak linealak, A.1 problema optimizazio koadratikoa dela esaten da:

$$\begin{aligned} \min f(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{s.a.} \quad g_i(\mathbf{x}) &= \mathbf{a}_i^T\mathbf{x} + b_i \leq 0, \quad i = 1,\dots,k, \\ h_i(\mathbf{x}) &= \mathbf{d}_i^T\mathbf{x} + e_i = 0, \quad i = 1,\dots,l, \end{aligned} \tag{A.5}$$

non $\mathbf{x}$, $\mathbf{c}$, $\mathbf{a}_i$ eta $\mathbf{d}_i \in \mathbb{R}^d$ bektoreak, $Q \in \mathbb{R}^{m\times m}$ matrizea positiboki erdi definituta eta $b_i$ eta $e_i$ konstanteak diren. Bi murrizketa daudenez, Lagrangeren bi biderkatzaile ditugu ($\alpha_i$ eta $\beta_i$); beraz, Lagrangeren funtzioa honela geratzen da:

$$\mathcal{L}(\mathbf{x},\alpha,\beta) = f(\mathbf{x}) + \sum_{i=1}^{k}\alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{l}\beta_i h_i(\mathbf{x}), \tag{A.6}$$

non $\alpha = (\alpha_1,\dots,\alpha_k)^T$, $\alpha_i \geq 0 \quad i = 1,\dots,k$ eta $\beta = (\beta_1,\dots,\beta_l)^T$. Optimizazio problema koadratikoa denean, Karush-Kuhn-Tucker-en teoremaren kasu partikular gisa, korolario hau ondorioztatzen da:

**A.0.5. korolarioa.** Izan bedi optimizazio koadratikoaren problema, A.5, haren soluzio optimoa $(\mathbf{x}^*,\alpha^*,\beta^*)$ existitzen da baldin eta eta eta soilik baldintza hauek betetzen badira:

$$\begin{gathered} \frac{\partial f(\mathbf{x}^*)}{\partial x_j} + \sum_{i=1}^{n}\alpha_i\frac{\partial g_i(\mathbf{x}^*)}{\partial x_j} + \sum_{i=1}^{n}\beta_i\frac{\partial h_i(\mathbf{x}^*)}{\partial x_j} = 0, \quad j = 1,\dots,m, \\ \alpha_i^* g_i(\mathbf{x}^*) = 0, \quad i = 1,\dots,k, \\ \alpha_i^* \geq 0, \quad i = 1,\dots,k, \\ h_i(\mathbf{x}^*) = 0, \quad i = 1,\dots,l. \end{gathered}$$

A.5 problema koadratikoa bere dualera aldatu ondoren, programazio koadratikoko teknikak erabiliz ebatzi egiten da. Xehetasun gehiago nahi izanez gero, kontsultatu: [46]

# B. Eranskina

# Kodea

## B.1 Datu basea

**Datu basea kargatu**

Kaggle webgunetik hartutako datu basea kargatuko dugu: https://www.kaggle.com/competitions/quora-insincere-questions-classification

[ ]:
```
#! pip install -q kaggle
# from google.colab import files
# files.upload()
#! mkdir ~/.kaggle
#! cp kaggle.json ~/.kaggle/
#! chmod 600 ~/.kaggle/kaggle.json
#! kaggle competitions download -c␣
↪quora-insincere-questions-classification
```

[ ]:
```
#! unzip 'quora-insincere-questions-classification.zip'
```

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
dir = "/content/drive/MyDrive/TFG/"
#!cp train.csv test.csv "/content/drive/MyDrive/TFG/"
```

```
Mounted at /content/drive
```

[2]:
```
import pandas as pd # Datuen analisirako prestatuta dago␣
↪(datuak ordenatu, banatu...), baita fitxategiak␣
↪kargatzeko ere.

df = pd.read_csv(dir + 'train.csv')
```

```
#Honakoa da gure dataframe: quid (erabiltzailea), question_text (galdera) eta target (0-Ez Toxikoa eta 1-Toxikoa)

df
```

[2]:

```
                         qid  \
0        00002165364db923c7e6
1        000032939017120e6e44
2        0000412ca6e4628ce2cf
3        000042bf85aa498cd78e
4        0000455dfa3e01eae3af
...                      ...
1306117  ffffcc4e2331aaf1e41e
1306118  fffed431801e5a2f4861
1306119  fffed48fb36b63db010c
1306120  ffffec519fa37cf60c78
1306121  ffffed09fedb5088744a

                                                question_text  target
0        How did Quebec nationalists see their province...       0
1        Do you have an adopted dog, how would you enco...       0
2        Why does velocity affect time? Does velocity a...       0
3        How did Otto von Guericke used the Magdeburg h...       0
4        Can I convert montra helicon D to a mountain b...       0
...                                                    ...     ...
1306117  What other technical skills do you need as a c...       0
1306118  Does MS in ECE have good job prospects in USA ...       0
1306119                          Is foam insulation toxic?       0
1306120  How can one start a research project based on ...       0
```

```
1306121  Who wins in a battle between a Wolverine and a...
↪    0

[1306122 rows x 3 columns]
```

[3]:
```
y = list(df['target'])

len(df),len(y),sum(y),sum(y)/len(y)*100

# 1306122 mezu daude guztira eta horietatik, bakarrik 80810
↪dira toxikoak. Hau da, mezuen %6-a soilik dira toxikoak,
↪beraz, gure datu basea guztiz desorekatuta dago.
```

[3]:
```
(1306122, 1306122, 80810, 6.187017751787352)
```

**Testuaren garbiketa**

Lehenik, mezu edo galdera errepikaturik edo hutsik dagoen ikusiko dugu.

[4]:
```
duplikatuta = df['question_text'].duplicated().sum()
hutsa = df['question_text'].isnull().sum()
print(duplikatuta, hutsa)
```

```
0 0
```

Jarraian, *gensim.utils.simple_preprocess* erabiliko dugu. Funtzio honek zerrenda batean itzuliko dizkigu gure datu basearen hitzak. Hitz hauei, *token* izena emango dizkiegu. Gainera, hitzak minuskuletara pasatuko ditu, puntuazio markak eta zenbakiak ezabatu...

[5]:
```
import gensim # Testua prozesatzeko tresna erabilgarriak
↪eskaintzen ditu.

from gensim.utils import simple_preprocess
#help(gensim.utils.simple_preprocess)
```

[6]:
```
df['question_text'] = df['question_text'].
↪apply(simple_preprocess)
df['question_text']
```

[6]:
```
0          [how, did, quebec, nationalists, see, their, p...
1          [do, you, have, an, adopted, dog, how, would, ...
2          [why, does, velocity, affect, time, does, velo...
3          [how, did, otto, von, guericke, used, the, mag...
4          [can, convert, montra, helicon, to, mountain, ...
                                ...
```

```
1306117    [what, other, technical, skills, do, you, need...
1306118    [does, ms, in, ece, have, good, job, prospects...
1306119                        [is, foam, insulation, toxic]
1306120    [how, can, one, start, research, project, base...
1306121    [who, wins, in, battle, between, wolverine, an...
Name: question_text, Length: 1306122, dtype: object
```

REMOVING STOPWORDS: Testuan erabilera nabarmenik ez duten hitzen ezabaketa da, hala nola, 'a', 'about', 'is', 'do'... Horretarako, *STOPWORDS* erabiliko dugu.

[7]:
```
from gensim.parsing.preprocessing import STOPWORDS
def stop_words (words):
  return [token for token in words if token not in STOPWORDS]

df['question_text'] = df['question_text'].apply(stop_words)
df['question_text']
```

[7]:
```
0                    [quebec, nationalists, province, nation]
1               [adopted, dog, encourage, people, adopt, shop]
2            [velocity, affect, time, velocity, affect, spa...
3                [otto, von, guericke, magdeburg, hemispheres]
4            [convert, montra, helicon, mountain, bike, cha...
                                ...
1306117          [technical, skills, need, science, undergrad]
1306118      [ms, ece, good, job, prospects, usa, like, ind...
1306119                              [foam, insulation, toxic]
1306120      [start, research, project, based, biochemistry...
1306121                        [wins, battle, wolverine, puma]
Name: question_text, Length: 1306122, dtype: object
```

LEMMATIZATION: Hitzak haren oinarrira bueltatzen ditu (Hiztegian aukitzen diren moduan).

[8]:
```
import nltk # Hizkuntzaren prozesamendurako liburutegi␣
 ↪ahaltsuenetakoa da. Horrekin batera, testuak prozesatzeko␣
 ↪liburutegi multzo bat ere eskaintzen du.

from nltk.stem import WordNetLemmatizer

nltk.download('wordnet')

lemmatizer = nltk.stem.WordNetLemmatizer()
```

```
def lemmatize_text(words):
  return [lemmatizer.lemmatize(w) for w in words]

df['question_text'] = df['question_text'].
 ↪apply(lemmatize_text)
df['question_text']
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

[8]:
```
0                    [quebec, nationalist, province, nation]
1               [adopted, dog, encourage, people, adopt, shop]
2          [velocity, affect, time, velocity, affect, spa...
3               [otto, von, guericke, magdeburg, hemisphere]
4          [convert, montra, helicon, mountain, bike, cha...
                                ...
1306117         [technical, skill, need, science, undergrad]
1306118    [m, ece, good, job, prospect, usa, like, india...
1306119                           [foam, insulation, toxic]
1306120    [start, research, project, based, biochemistry...
1306121                        [win, battle, wolverine, puma]
Name: question_text, Length: 1306122, dtype: object
```

Bukatzeko, datuak testu gisa utziko ditugu.

[9]:
```
X = [ ' '.join(words) for words in df['question_text']]
```

**Datuen analisia:**

HITZEN ANALISIA (X)

Atal honetan gure datu basean gehien errepikatzen diren hitzak zeintzuk diren aztertuko dugu. Horretarako, jarraian agertzen diren bi grafikoak irudikatu ditugu:

Histograma funtzioa sortuko dugu:

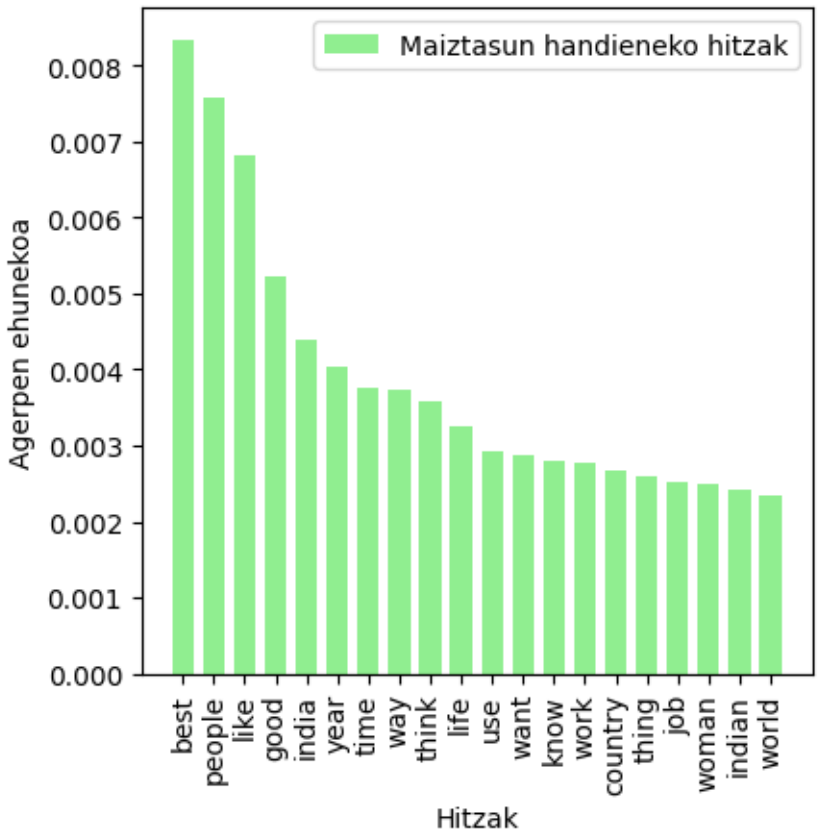
[ ]:
```
# Gako bezala hitzak eta balio bezala hitz bakoitzari␣
 ↪dagokion maiztasuna dituen hiztegia sortzen da.
def histograma (X):
  hist = dict()
  for txt in X:
    for w in txt.split():
      if w not in hist:
        hist[w] =1
      else:
        hist[w]+=1
  norm = sum(hist.values())
  for g,b in hist.items():
```

```
    hist[g]=b/norm
  return hist
```

[ ]:
```
histo = histograma(X)
```

[ ]:
```
import matplotlib.pyplot as plt # Grafikoak sortzeko␣
↪erabiltzen da.
import numpy as np #
import operator # Funtzio multzi bat da, hizkuntzarekin␣
↪eragiketak imitatzen dutenak.

n = 20 # Hitzen kopurua adierazten du.
hitzak =[] # Hitzak gordetzeko zerrenda hutsa da.
maiztasuna =[] # Hitzen agerpen-kopurua gordetzeko zerrenda␣
↪hutsa da.
x_long = np.arange(n) # Hitzen indizeak ordezkatzen dituen␣
↪NumPy sorta da.
for g,b in sorted(histo.items(), key = operator.
↪itemgetter(1),reverse= True)[:n]: # Histogramaren␣
↪elementuak hitzak eta maiztasun indexatzen dira. (g, b)␣
↪hitzak eta bere agerpen-kopurua ditu. Hitzak agerpenaren␣
↪arabera ordenatzen dira eta n hitzek mugatzen dute.
  hitzak.append(g)
  maiztasuna.append(b)
fig , ax = plt.subplots(figsize = [4.5,4.5]) #␣
↪Matplotlib-eko irudi eta ardatz berria tamainua zehaztuta␣
↪sortzen du.
ax.bar(x_long, maiztasuna, width =0.7, label = 'Maiztasun␣
↪handieneko hitzak', color= 'lightgreen') # Barra grafikoa␣
↪sortzen du emandako datuekin.
ax.set_ylabel('Agerpen ehunekoa') # y ardatzaren etiketa␣
↪ezartzen du.
ax.set_xlabel('Hitzak') # x ardatzaren etiketa ezartzen du.
ax.set_xticks(x_long) # x ardatzaren hitzen posizioa␣
↪zehazten du.
ax.set_xticklabels(hitzak, rotation=90)
ax.legend() # Legenda gehitzen du.
```

[ ]:
```
<matplotlib.legend.Legend at 0x7889108fc6a0>
```




WORD CLOUD: Mezuetan gehien errepikatzen diren hitzen irudia da.

[ ]:
```
from wordcloud import WordCloud # Pythoneko WordCloud liburutegia, hitz-hodeiak sortzeko erabiltzen da.

plt.figure(figsize=(7.5, 5)) # Irudiaren tamaina da.

wc = WordCloud(max_words=200, width=800, height=600, background_color='white', colormap='gist_rainbow').generate(" ".join(X)) # Irudikatuko ditugun hitzen multzoa da, hitzak ondo ikusteko hitz kopurua eta tamaina zehaztuko dugu.

plt.imshow(wc, interpolation='bilinear') # wc (hitz multzoa) irudi gisa erakutsiko du, interpolazio bilineala erabiliz ikus-kalitatea hobetzeko.
```

```
plt.axis('off')  # Ardatzak ez agertzeko.

plt.show() # Irudia agertzeko.
```


### Datuen banaketa

[10]:
```
from sklearn.model_selection import train_test_split #␣
↪Scikit-learn-ek datuak banatzeko ahalbidetzen duen␣
↪funtzioa da.

X_train,X_test,y_train,y_test = train_test_split(X,y,␣
↪test_size=0.2) #Train (Entrenamendua) hasierako datuen %␣
↪80 da.

X_val,X_test,y_val,y_test = train_test_split(X_test,y_test,␣
↪test_size=0.5) #Test (Testa) hasierako datuen % 10 da.␣
↪Validation (Balidazioa) hasierako datuen % 10 da.
```

[11]:
```
#Ikus dezagun mezuen kopurua banaketa egin ostean: 1044897␣
↪mezu train-ean, 130612 validation eta 130613 test-ean.

for x in (X_train,X_val,X_test,y_train,y_val,y_test):
```

```
  print(f'{len(x)}')
```

```
1044897
130612
130613
1044897
130612
130613
```

### Google Driven gorde

[12]:
```
import pickle # Informazioa fitxategi bitarretan era␣
 ↪bizkorrean gordetzeko erabiltzen da.
#help(pickle.dump)

with open(dir + "data.bin",mode="wb") as f:
  pickle.dump((X_train,X_val,X_test,y_train,y_val,y_test),f)
```

## B.2 Naive Bayes

### Google drivetik kargatu

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
import pickle
with open("/content/drive/MyDrive/TFG/data.bin",mode="rb")␣
↪as f:
  X_train,X_val,X_test,y_train,y_val,y_test = pickle.load(f)
```

### Eredua sortzen

Lehenik, maiztasun hiztegia funtzioa sortuko dugu zeinetan testu bat hartuta hitza eta haren maiztasunarekin hiztegi bat bueltatzen duen.

Funtzio horren laguntzaz, gure lehen eredua definitzen dugu. Toxikoak diren hiztegi bat, toxikoak ez diren hiztegi bat eta haien proportzioak bueltatzen dituena.

[3]:
```
def maiztasun_hiztegia (X, n):

  hiztegia = {} # hiztegi hutsa

  for text in X: # X-ko hitz guztiak hiztegiko gakoak izango␣
↪dira eta balio bezala hitz bakoitzari dagokion agerpen␣
↪kopurua.
    for word in text.split():
      hiztegia [word] = hiztegia.get (word, 0) + 1

  minhiztegia = min(hiztegia.values()) # Agerpen guztietatik␣
↪txikiena hartuko dugu.
  hiztegia ['#OOV#']= minhiztegia / n # Out Off Vocabulary␣
↪hitza gehituko dugu balio oso txikia esleituz.

  norm = sum(hiztegia.values()) # Agerpen guztiak zenbatuko␣
↪dira.

  for g,b in hiztegia.items(): # Agerpenak agerpen kopuru␣
↪guztietatik zatitzen dira, maiztasunak lortzeko.
    hiztegia[g] = b / norm

  return hiztegia # Maiztasun hiztegia itzultzen da.
```

[4]:
```
def MyModel(X,y,n = 1e10):

  X0 = [] # Zerrenda hutsak
  X1 = []

  for text, label in zip(X, y): # X-ko elementuak klasearen␣
↪arabera zerrendara gehituko dira.
    if label == 0:
      X0.append(text)
    else:
      X1.append(text)

  h0 = maiztasun_hiztegia(X0,n) # Maiztasun hiztegiak␣
↪sortzen dira.
  h1 = maiztasun_hiztegia(X1,n)

  return h0,h1,len(X0)/len(X),len(X1)/len(X) # Klase␣
↪bakoitzaren maitasun hiztegiak eta klase bakoitzaren␣
↪proportzioa itzultzen da.
```

[5]:
```
model = MyModel(X_train,y_train,1e10)
```

[6]:
```
# Ikus dezagun hitz kopurua klasearen arabera zenbatekoa den.
 ↪ Espero dugu h0-n dauden hitz kopurua handiagoa izatea, 0␣
 ↪klaseako (ez toxikoak) mezu gehiago baitaude.
h0,h1,_,_ = model

len(h0),len(h1)
```

[6]:
```
(143689, 29093)
```

**Eredua ebaluatzen**

Gure eredua ebaluatuko dugu. Horretarako, lehenik, sortutako ereduaren balioekin etiketak estimatuko ditugu, Naive Bayes algoritmoa jarraituz. Ondoren, ebaluazioan azaldutako hainbat metrika lortuko ditugu.

[8]:
```
from math import log

def logprob(x,h,p):

  m_oov = h['#OOV#'] # OOV-ko hitzei esleitutako maiztasuna␣
↪lortzen du.
```

```
  m = [log(h.get(w,m_oov)) for w in x.split()] # Hitz␣
↪bakoitzaren maiztasunaren logaritmoa kalkulatzen da.

  return sum(m) + log(p) # Hitz guztien maiztasunaren␣
↪logaritmoak eta probabilitatearen logaritmoak batzen dira.
```

[9]:
```
# Azaldu berri den funtzioa exekutatuko da klase␣
↪bakoitzarentzat. Input-ak eredutik hartuko dira.

def logprobs(x,model):

  h0,h1,p0,p1 = model

  return logprob(x,h0,p0) , logprob(x,h1,p1)
```

[10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix, roc_curve,␣
↪auc, classification_report, precision_recall_curve
from math import exp

def ebaluazioa (X, y, model, full=False):

    lprob = [logprobs(x, model) for x in X] # Klase␣
↪bakoitzari dagokion probabilitateen zerrenda.
    prob = [z/(1+z) for z in (exp(p1-p0) for p0, p1 in␣
↪lprob)] # Probabilitatea zenbatekoa den.
    #prob = [p1-p0 for p0, p1 in lprob] # Probabilitatea␣
↪zenbatekoa den.
    pred = [(0 if p0 >= p1 else 1) for p0, p1 in lprob] #␣
↪Zerrendatik klaseak antzematu ditu, probabilitate␣
↪handiena hartuz.

    #Sailkatze matrizea irudikatzeko:
    conf_matrix = confusion_matrix(y, pred)

    # ROC kurba eta AUC-a irudikatzeko:
    fpr, tpr,_ = roc_curve(y, prob)
    roc_auc = auc(fpr, tpr)

    # Precisión-Recall kurba irudikatu:
```

```
    precision, recall, _ = precision_recall_curve(y, prob)
    pr_auc= auc(recall,precision)

    if full :
      plt.figure(figsize=(9,3))
      plt.subplot(1, 3, 1)
      sns.heatmap(conf_matrix, annot=True, fmt="d",␣
↪cmap='Blues')
      plt.title('Sailkatze matrizea (c = 0.5)')
      plt.ylabel('Benetako balioak')
      plt.xlabel('Estimatutako balioak')

      plt.subplot(1, 3, 2)
      plt.plot(fpr, tpr, color='lightskyblue', lw=2,␣
↪label='ROC AUC = %0.3f' % roc_auc)
      plt.plot([0, 1], [0, 1], color='darkblue', lw=2,␣
↪linestyle='--')
      plt.xlabel('1 - Espezifikotasuna')
      plt.ylabel('Sentikortasuna')
      plt.title('ROC kurba')
      plt.legend(loc="lower right")

      plt.subplot(1, 3, 3)
      plt.plot(recall, precision, color='lightskyblue',␣
↪lw=2, label='PR AUC = %0.3f)' % pr_auc)
      plt.axhline(y = sum(y) / len(y), color='darkblue',␣
↪linestyle='--')
      plt.xlabel('Recall')
      plt.ylabel('Precision')
      plt.title('Precision-Recall kurba')
      plt.legend(loc="upper right")
      plt.tight_layout()
      plt.show()

      print('\n', classification_report(y, pred)) # Hainbat␣
↪metriken balioak taula batean itzultzen ditu.
      print('accuracy =', sum(y2==y for y,y2 in zip(y,pred))␣
↪/ len(X) ) # Egoki sailkatu diren etiketen proportzioa,␣
↪hau da, accuracy itzultzen du.

    print('PR AUC =', pr_auc)
```

**Probak egiten**

Balidazioko datu multzoa erabiliz, n parametroa finkatuko dugu. Horretarako, balio desberdinak emango dizkiogu eta AUC altuena duen eredua aukeratuko dugu.

[11]:
```
model_2 = MyModel(X_train,y_train,1e-2)
ebaluazioa(X_val, y_val, model_2)
```

```
PR AUC = 0.2840040843076209
```

[12]:
```
model0 = MyModel(X_train,y_train,1e0)
ebaluazioa(X_val, y_val, model0)
```

```
PR AUC = 0.504135074105791
```

[19]:
```
model1 = MyModel(X_train,y_train,1e1)
ebaluazioa(X_val, y_val, model1)
```

```
PR AUC = 0.5134742935124083
```

[14]:
```
model2 = MyModel(X_train,y_train,1e2)
ebaluazioa(X_val, y_val, model2)
```

```
PR AUC = 0.505128381672088
```

[15]:
```
model3 = MyModel(X_train,y_train,1e3)
ebaluazioa(X_val, y_val, model3)
```

```
PR AUC = 0.493542938494174
```

[16]:
```
model4 = MyModel(X_train,y_train,1e4)
ebaluazioa(X_val, y_val, model4)
```

```
PR AUC = 0.4828431605852408
```

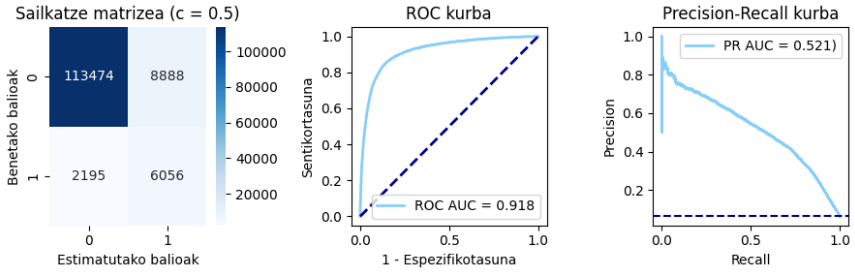
[17]:
```
model6 = MyModel(X_train,y_train,1e6)
ebaluazioa(X_val, y_val, model6)
```

```
PR AUC = 0.46698128604923606
```

Ikusten dugu hasieran n handitu ahala AUC-aren balioa handiagoa dela, aldiz, n = 1e1 tik aurrera, berriz txikitzen da. Gogoratu n-ak hiztegian ez dauden hitzen maiztasuna adierazten duela. Orokorrean, oso antzekoak dira emaitz guztiak, baina altuena n = 1e1 denean lortzen da. Beraz, eredu horrekin geratuko gara.

**Emaitzak**

[18]:
```
ebaluazioa(X_test, y_test, model1,True)
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.93 | 0.95 | 122362 |
| 1 | 0.41 | 0.73 | 0.52 | 8251 |
| accuracy | | | 0.92 | 130613 |
| macro avg | 0.69 | 0.83 | 0.74 | 130613 |
| weighted avg | 0.94 | 0.92 | 0.93 | 130613 |

```
accuracy = 0.9151462718106161
PR AUC = 0.5211487962472399
```

## B.3 Bektorizazioa

### Google Drivetik kargatu

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
dir = "/content/drive/MyDrive/TFG/"
```

```
Mounted at /content/drive
```

[2]:
```
import pickle # Fitxategia irekitzeko erabiltzen da.

with open(dir + "data.bin",mode="rb") as f:
  X_train,X_val,X_test,y_train,y_val,y_test = pickle.load(f)
```

### Bektorizazioa

[3]:
```
print(type(X_train))
```

```
<class 'list'>
```

[4]:
```
def indize_hiztegia(X,n=1000):

  h = {} # Hiztegi hutsa

  for text in X: # X-ko hitz guztiak hiztegiko gakoak izango␣
↪dira eta balio bezala hitz bakoitzari dagokion agerpen␣
↪kopurua.
    for word in text.split():
      h [word] = h.get (word, 0) + 1

  z = sorted(h.items(), key=lambda item: item[1],reverse=␣
↪True) # Zerrenda bat da, hitza eta haren agerpenaren␣
↪tuplak osatzen dutenak. Gainera, agerpen kopuru␣
↪handienetik txikienera ordenatuta dago.

  return {g:i for i,(g,b) in enumerate(z[:n])} # Funtzio␣
↪honek beste hiztegi bat bueltatuko du non zerrendako␣
↪hitzak gakoak diren eta balioa hitzaren posizioa zerendan.
↪ Hitz kopurua mugatzeko n balioa erabiltzen da.
```

[5]:
```
#Ikus dezagun zeintzuk diren 'train' datuekin lortzen dugun␣
↪lehenego 10 hitzetako indize hiztegia.
#word2index = indize_hiztegia(X_train,10)
#word2index
```

[6]:
```
# Gure entrenamenduko datuetan dauden hitz desberdin guztien␣
↪kopurua kalkulatuko dugu, hau da, gure 'bokabularioa'.
voc = {w for txt in X_train for w in txt.split()}
len(voc)
```

[6]:
```
148468
```

[7]:
```
word2index = indize_hiztegia(X_train,len(voc))
```

Gure datu multzoa nahiko handia denez, memoria arazoak izan ditzakegu datuak bektorizatzerakoan. Ondorioz, matrize dispertsoak erabili ditugu, hauek zero ez diren elementuak bakarrik gordetzen dituzte, memoriaren erabilera optimizatuz eta zero asko dituzten datu-multzo handietan eragiketa azkarragoak egiteko aukera ematen dute.

[8]:
```
from scipy.sparse import lil_matrix #lil_matrix motako␣
↪matrizeak erabiliko ditugu memoriagatik

def vectorize(X,hizt):

  n = len(X) # Matrizearen errenkada kopurua.
  m = len(hizt) + 1 # Matrizearen zutabe kopurua (+ 1 'out␣
↪off vocabulary' adierazten du.)

  M = lil_matrix((n,m),dtype=np.int64) # SciPy motako n x m␣
↪-ko M matrizea sortzen da.

  for i,text in enumerate(X): # M agerpen matrizea da, X-ko␣
↪eta hiztegiko hitzez sortua.
    for word in text.split():
      M[i,hizt.get(word,-1)] += 1

  return M.tocsr() # Compressed Sparse Row array. Matrize␣
↪hau egokiagoa da eredu aljebraikoekin lan egiteko.
```

[9]:
```
import numpy as np

# Mezuak bektorizatuko ditugu.

X_train = vectorize(X_train, word2index)
X_val = vectorize(X_val, word2index)
X_test = vectorize(X_test, word2index)
```

```
# Haien tamaina ikusiko dugu.
#print(X_train.shape,X_val.shape,X_test.shape)

#Matrizearen azken zutabean dauden elementu guztien batura␣
↪kalkulatuko dugu.
#print(X_train[:,-1].sum() , X_val[:,-1].sum() , X_test[:
↪,-1].sum())
```

[10]:
```
print(type(X_train))
```

```
<class 'scipy.sparse._csr.csr_matrix'>
```

[11]:
```
# Etiketak bektorizatuko ditugu. Nahikoa da NumPy-ko 'array'␣
↪bihutzea.

y_train = np.asarray(y_train)
y_val = np.asarray(y_val)
y_test = np.asarray(y_test)
```

**Google driven gorde**

[12]:
```
import pickle # Bektorizatutako datuak gordeko ditugu,␣
↪fitxategi berri batean.

with open(dir + "vdata.bin", "wb") as f:
  pickle.dump((X_train,X_val,X_test,y_train,y_val,y_test),f)
```

## B.4 Bektoreen bidezko eredua

### Google drivetik kargatu

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
dir = "/content/drive/MyDrive/TFG/"
```

```
Mounted at /content/drive
```

[2]:
```
import pickle # Fitxategia irekitzeko erabiltzen da.

with open(dir + "vdata.bin",mode="rb") as f:
  X_train,X_val,X_test,y_train,y_val,y_test = pickle.load(f)
```

**Eredua sortzen**

Naive Bayes ereduaren baliokidea lortzea da gure helburua, baina oraingoan matrizeak erabiliz.

[3]:
```
import numpy as np # Array-ak sortzeko eta bektore eta␣
↪matrizeekin lan egitea ahalbidetzen du.

def MyModel(X,y,n = 1e1):

  agerpen_minimoa = X[:,:-1].sum(axis=0).min()

  X0 = X[y == 0,:] # X0 X matrizeko errenkada guztiak izango␣
↪ditu non y-ren etiketak 0 diren.
  agerpenak0 = X0.sum(axis=0) * 1.0 # Zutabeak batzen dira,␣
↪hau da, hitzen agerpenak.
  batura0 = agerpenak0.sum() + 1/n
  agerpenak0[agerpenak0 == 0] = agerpen_minimoa/n # Hitz bat␣
↪0 balio izatekotan, 0 ez den balio oso txikia esleituko␣
↪zaio.
  maiztasunak0 = agerpenak0 / batura0 # Zatiketa honen␣
↪bidez, agerpenak ordez, maiztasunak lortzen dira.
  print(np.log(maiztasunak0))

  X1 = X[y == 1,:] # Berdin egiten da etiketak 1 direnean.
  print(X1.shape)
  agerpenak1 = X1.sum(axis=0) * 1.0
  batura1 = agerpenak1.sum() + 1/n
  agerpenak1[agerpenak1 == 0] = agerpen_minimoa/n
  maiztasunak1 = agerpenak1 / batura1
```

```
  print(np.log(maiztasunak1))

  V = np.log(np.concatenate((maiztasunak0, maiztasunak1))) #␣
 ↪Maiztasun bektorea lortzen dugu (lehenengo errenkada␣
 ↪etiketak 0 dituztenak eta bigarrenak etiketak 1␣
 ↪dituztenak) eta logaritmoa aplikatzen diogu balio␣
 ↪bakoitzari.
  lp = np.log(np.asarray([X0.shape[0]/X.shape[0] , X1.
 ↪shape[0]/X.shape[0]])) # Etiketa 0 eta 1 izateko␣
 ↪probabilitateen logaritmoen matrizea da hurrenez hurren.

  return V,lp
```

[4]:
```
# Entrenamenduko datuekin gure eredua sortuko dugu.
model = MyModel(X_train, y_train, 1e1)
model
```

```
[[ -4.70383506  -5.04877893  -5.0240731  ... -15.51362361 -15.
 ↪51362361
  -17.81620871]]
(64571, 148469)
[[ -6.91326037  -3.91724133  -4.6713716  ... -15.47542692 -15.
 ↪47542692
  -15.47542692]]
```

[4]:
```
(matrix([[ -4.70383506,  -5.04877893,  -5.0240731 , ..., -15.
 ↪51362361,
          -15.51362361, -17.81620871],
         [ -6.91326037,  -3.91724133,  -4.6713716 , ..., -15.
 ↪47542692,
          -15.47542692, -15.47542692]]),
 array([-0.06378843, -2.7839082 ]))
```

### Eredua ebaluatzen

[5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix, roc_curve,␣
 ↪auc, classification_report, precision_recall_curve,␣
 ↪average_precision_score
from math import exp
from scipy.sparse import csr_matrix
```

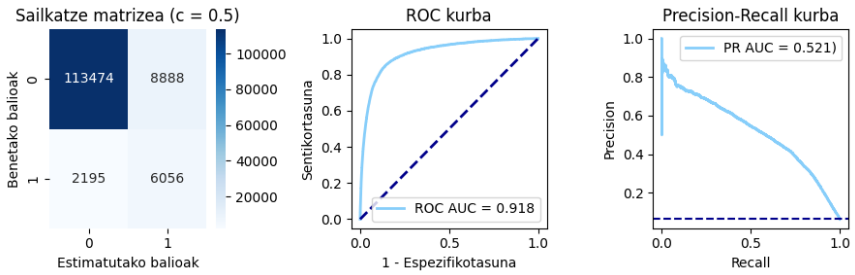
```
def ebaluazioa (X, y, model, full=False):

    V,lp = model # Ereduaren matrizeak lortu.
    V = csr_matrix(V) # csr.matrix motakoa bihurtu, ondoren␣
↪eragiketak ondo egiteko.
    logprobs = np.asarray(X @ V.T + lp) # Gure ereduak␣
↪estimatzen dituen erantzunak.

    pred = np.argmax(logprobs, axis=1) # Predikzioak, balio␣
↪handiena aukeratuz.
    score = logprobs[:,1]-logprobs[:,0] # Probablitateak␣
↪lortu.

    #Sailkatze matrizea irudikatzeko:
    conf_matrix = confusion_matrix(y, pred)

    # ROC kurba eta AUC-a irudikatzeko:
    fpr, tpr,_ = roc_curve(y, score)
    roc_auc = auc(fpr, tpr)
    # Precisión-Recall kurba eta AUC-a irudikatu:
    precision, recall, _ = precision_recall_curve(y, score)
    pr_auc= auc(recall,precision)

    if full :
      plt.figure(figsize=(9,3))
      plt.subplot(1, 3, 1)
      sns.heatmap(conf_matrix, annot=True, fmt="d",␣
↪cmap='Blues')
      plt.title('Sailkatze matrizea (c = 0.5)')
      plt.ylabel('Benetako balioak')
      plt.xlabel('Estimatutako balioak')

      plt.subplot(1, 3, 2)
      plt.plot(fpr, tpr, color='lightskyblue', lw=2,␣
↪label='ROC AUC = %0.3f' % roc_auc)
      plt.plot([0, 1], [0, 1], color='darkblue', lw=2,␣
↪linestyle='--')
      plt.xlabel('1 - Espezifikotasuna')
      plt.ylabel('Sentikortasuna')
      plt.title('ROC kurba')
      plt.legend(loc="lower right")
```

```
    plt.subplot(1, 3, 3)
    plt.plot(recall, precision, color='lightskyblue',␣
↪lw=2, label='PR AUC = %0.3f)' % pr_auc)
    plt.axhline(y = sum(y) / len(y), color='darkblue',␣
↪linestyle='--')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Precision-Recall kurba')
    plt.legend(loc="upper right")
    plt.tight_layout()
    plt.show()

    print('\n', classification_report(y, pred)) # Hainbat␣
↪metriken balioak taula batean itzultzen ditu.
    print('accuracy =', sum(y == np.
↪argmax(logprobs,axis=1)) / len(y)) # Egoki sailkatu diren␣
↪etiketen proportzioa, hau da, accuracy itzultzen du.

  print('PR AUC =', pr_auc)
```

[6]:
```
ebaluazioa(X_val, y_val, model)
```

```
PR AUC = 0.5134742935124083
```

[7]:
```
ebaluazioa(X_test, y_test, model, True)
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.93 | 0.95 | 122362 |
| 1 | 0.41 | 0.73 | 0.52 | 8251 |

```
    accuracy                           0.92    130613
   macro avg       0.69      0.83      0.74    130613
weighted avg       0.94      0.92      0.93    130613

accuracy = 0.9151462718106161
PR AUC = 0.5211487970947268
```

Emaitzak Naive Bayes ereduaren berdin-berdinak dira. Bektoreen bidez eredu baliokidea lortu dugu.

## B.5 Erregresio Logistikoa

### Datuak kargatu

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
dir = "/content/drive/MyDrive/TFG/"
```

```
Mounted at /content/drive
```

[2]:
```
import pickle
with open(dir + "vdata.bin",mode="rb") as f:
  X_train,X_val,X_test,y_train,y_val,y_test = pickle.load(f)
```

### Eredua sortzen

Erregresio logistikoko eredua sortzeko, Pythonek eskaintzen duen *'LogisticRegression'* funtzioa erabiliko dugu. Honek parametro desberdinak har ditzake, haien artean: * C: defektuz = 1.0 (erregularizazioa kontrolatzeko) * solver : {'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'} (Optimizazio-probleman erabili beharreko algoritmoa) * max_iter : defektuz=100 (hartutako iterazioen gehieneko kopurua.)

[3]:
```
import numpy as np

from sklearn.linear_model import LogisticRegression #␣
↪Erregresio logistikoko paketea kargatzen da.

def model_l_r(X, y, C=1, solver='lbfgs', max_iter = 100):
    model = LogisticRegression(C=C, solver=solver,␣
↪max_iter=max_iter)
    model.fit(X, y)
    return model
```

### Eredua ebaluatzen

[4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix, roc_curve,␣
↪auc, classification_report, precision_recall_curve

def ebaluazioa (X, y, model, full=False):

    # X-ren klaseak aurresango ditu:
    y_pred = model.predict(X)
```

```
    #Sailkatze matrizea irudikatzeko:
    conf_matrix = confusion_matrix(y, y_pred)

    # 1 klasearen probabilitateak lortzen ditu:
    y_score = model.predict_proba(X)[:, 1]

    # ROC kurba eta AUC-a irudikatzeko:
    fpr, tpr, _ = roc_curve(y, y_score)
    roc_auc = auc(fpr, tpr)


    # Precisión-Recall kurba eta AUC-a irudikatu:
    precision, recall, _ = precision_recall_curve(y, ␣
↪y_score)
    pr_auc= auc(recall,precision)

    if full :
      plt.figure(figsize=(9,3))
      plt.subplot(1, 3, 1)
      sns.heatmap(conf_matrix, annot=True, fmt="d",␣
↪cmap='Blues')
      plt.title('Sailkatze matrizea (c = 0.5)')
      plt.ylabel('Benetako balioak')
      plt.xlabel('Estimatutako balioak')

      plt.subplot(1, 3, 2)
      plt.plot(fpr, tpr, color='lightskyblue', lw=2,␣
↪label='ROC AUC = %0.3f' % roc_auc)
      plt.plot([0, 1], [0, 1], color='darkblue', lw=2,␣
↪linestyle='--')
      plt.xlabel('1 - Espezifikotasuna')
      plt.ylabel('Sentikortasuna')
      plt.title('ROC kurba')
      plt.legend(loc="lower right")

      plt.subplot(1, 3, 3)
      plt.plot(recall, precision, color='lightskyblue',␣
↪lw=2, label='PR AUC = %0.3f)' % pr_auc)
      plt.axhline(y = sum(y) / len(y), color='darkblue',␣
↪linestyle='--')
      plt.xlabel('Recall')
```

```
        plt.ylabel('Precision')
        plt.title('Precision-Recall kurba')
        plt.legend(loc="upper right")
        plt.tight_layout()
        plt.show()

        print('\n', classification_report(y, y_pred))
        print('accuracy =', model.score(X,y))

    print('PR AUC =', pr_auc)
```

**Probak egiten**

Parametroak aldatuko ditugu eta balidazio multzoarekin ebaluatuko ditugu sortutako eredu desberdinak, emaitza hoberenak dituena aukeratzeko:

[5]:
```
# 'liblinear' erabiliz: ( egokiak datu multzo bitar eta␣
↪ertainetarako)

model = model_l_r(X_train, y_train,  C=1, solver =␣
↪'liblinear', max_iter=100) # Entrenamenduko datuekin gure␣
↪eredua sortuko dugu.
ebaluazioa(X_val, y_val, model) # Ebaluatzen, PR AUC␣
↪handiena lortzen duen eredua aukeratuko dugu.
```

```
PR AUC = 0.5838147730541734
```

[6]:
```
# Iterazio kopurua handitu.
model0 = model_l_r(X_train, y_train, C=1,␣
↪solver='liblinear', max_iter = 1000)
ebaluazioa(X_val, y_val, model0)
```

```
PR AUC = 0.5838147730541734
```

[7]:
```
# C balioa handitu.
model1 = model_l_r(X_train, y_train, C=7,␣
↪solver='liblinear', max_iter = 100)
ebaluazioa(X_val, y_val, model1)
```

```
PR AUC = 0.5681161426088135
```

[8]:
```
# Iterazio kopurua txikitu.
model2 = model_l_r(X_train, y_train, C=1,␣
↪solver='liblinear', max_iter = 50)
```

```
ebaluazioa(X_val, y_val, model2)
```

```
PR AUC = 0.5838147730541734
```

[9]:
```
# 'lbfgs' erabiliz: (datu multinomialetarako)

model3 = model_l_r(X_train, y_train, solver='lbfgs')
ebaluazioa(X_val, y_val, model3)
```

```
PR AUC = 0.5825861774470291

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
 ↪_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the␣
 ↪data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver␣
 ↪options:
    https://scikit-learn.org/stable/modules/linear_model.
 ↪html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[10]:
```
# Iterazio kopuru handiak.
model4 = model_l_r(X_train, y_train, solver='lbfgs',␣
 ↪max_iter=1000)
ebaluazioa(X_val, y_val, model4)
```

```
PR AUC = 0.5838165104817505
```

[11]:
```
# 'sag' erabiliz: (datu multzo handietarako azkarragoa)␣
 ↪Honek konbergentzia arazoak ditu.
model5 = model_l_r(X_train, y_train, solver='sag',␣
 ↪max_iter=50)
ebaluazioa(X_val, y_val, model5)
```

```
PR AUC = 0.5845303188574883

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
 ↪_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the␣
 ↪coef_ did not
```

```
converge
  warnings.warn(
```

[12]:
```
model6 = model_l_r(X_train, y_train,C=10, solver='sag')
ebaluazioa(X_val, y_val, model6)
```

```
PR AUC = 0.5815431323150492

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
 ↪_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the␣
 ↪coef_ did not
converge
  warnings.warn(
```

[13]:
```
model7 = model_l_r(X_train, y_train,C=1, solver='sag',␣
 ↪max_iter = 200)
ebaluazioa(X_val, y_val, model7)
```

```
PR AUC = 0.5839419658820861

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
 ↪_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the␣
 ↪coef_ did not
converge
  warnings.warn(
```

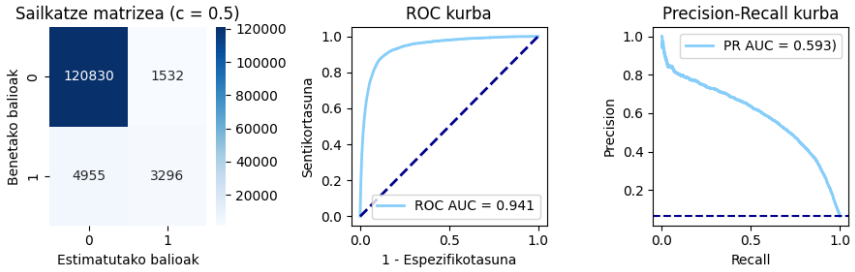
[14]:
```
model8 = model_l_r(X_train, y_train, solver='sag')
ebaluazioa(X_val, y_val, model8)
```

```
PR AUC = 0.5843731079421346

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
 ↪_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the␣
 ↪coef_ did not
converge
  warnings.warn(
```

Lortutako Precision-Recall kurbaren balioak oso antzekoak dira. Beraz, gure datuak bitarrak direnez eta denbora arazorik ez daudenez, 'liblinear' aukeratuko dugu, C = 1 eta gehigizko iterazioak = 100 aukeratuz (defektuzkoak).

**Emaitzak**

[15]:
```
ebaluazioa(X_test, y_test, model, True)
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.99 | 0.97 | 122362 |
| 1 | 0.68 | 0.40 | 0.50 | 8251 |
| accuracy | | | 0.95 | 130613 |
| macro avg | 0.82 | 0.69 | 0.74 | 130613 |
| weighted avg | 0.94 | 0.95 | 0.94 | 130613 |

```
accuracy = 0.9503341933804446
PR AUC = 0.5929073060075805
```

## B.6 Support Vector Machine

### Datuak kargatu

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
dir = "/content/drive/MyDrive/TFG/"
```

```
Mounted at /content/drive
```

[2]:
```
import pickle
with open(dir + "vdata.bin",mode="rb") as f:
  X_train,X_val,X_test,y_train,y_val,y_test = pickle.load(f)
```

### Eredua sortzen

*LinearSVC* (Support Vector Classifier sailkapen linealeko ezarpen eraginkor eta malgu bat da, datu-multzo handietara ondo egokitzen dena eta penalizazioen eta galera-funtzioen artean hautatzeko aukera ematen duena. LinearSVC-ak input dentsoak zein dispertsoak jasaten ditu, gure datuen kasuan nahiko baliagarria da informazio hau. Honek parametro desberdinak har ditzake, haien artean: * C: defektuz = 1.0 (erregularizazioa kontrolatzeko) * max_iter : defektuz=1000 (hartutako iterazioen gehieneko kopurua.)

Gehiago ikusteko : Scikit-learn - SVC

[3]:
```
from sklearn.svm import LinearSVC

def L_SVM (X, y, max_iter = 1000):
    model = LinearSVC(max_iter = max_iter)
    model.fit(X, y)
    return model
```

### Eredua ebaluatzen

[4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix, roc_curve,␣
 ↪auc, classification_report, precision_recall_curve
from sklearn.metrics import accuracy_score

def ebaluazioa (X, y, model, full=False):

    y_pred = model.predict(X)

    conf_matrix = confusion_matrix(y, y_pred)
```

```
    decision_scores = model.decision_function(X)

    # ROC kurba eta AUC-a irudikatzeko:
    fpr, tpr, _ = roc_curve(y, decision_scores)
    roc_auc = auc(fpr, tpr)

    # Precisión-Recall kurba irudikatu:
    precision, recall, _ = precision_recall_curve(y, ␣
↪decision_scores)
    pr_auc= auc(recall,precision)

    if full :
      plt.figure(figsize=(9,3))
      plt.subplot(1, 3, 1)
      sns.heatmap(conf_matrix, annot=True, fmt="d",␣
↪cmap='Blues')
      plt.title('Sailkatze matrizea (c = 0.5)')
      plt.ylabel('Benetako balioak')
      plt.xlabel('Estimatutako balioak')

      plt.subplot(1, 3, 2)
      plt.plot(fpr, tpr, color='lightskyblue', lw=2,␣
↪label='ROC AUC = %0.3f' % roc_auc)
      plt.plot([0, 1], [0, 1], color='darkblue', lw=2,␣
↪linestyle='--')
      plt.xlabel('1 - Espezifikotasuna')
      plt.ylabel('Sentikortasuna')
      plt.title('ROC kurba')
      plt.legend(loc="lower right")

      plt.subplot(1, 3, 3)
      plt.plot(recall, precision, color='lightskyblue',␣
↪lw=2, label='PR AUC = %0.3f)' % pr_auc)
      plt.axhline(y = sum(y) / len(y), color='darkblue',␣
↪linestyle='--')
      plt.xlabel('Recall')
      plt.ylabel('Precision')
      plt.title('Precision-Recall kurba')
      plt.legend(loc="upper right")
      plt.tight_layout()
      plt.show()
```

```
    print('\n', classification_report(y, y_pred))
    print('accuracy =', accuracy_score(y, y_pred))

  print('PR AUC =', pr_auc)
```

**Probak egiten**

Baliodazio datu-mutzoa erabiliz, gehiegizko iterazio kopurua aukeratuko dugu, eredua ahalik eta gehien hobetzeko.

[5]:
```
model = L_SVM (X_train, y_train, max_iter = 1000)
ebaluazioa (X_val, y_val, model)
```

```
PR AUC = 0.5661860799217673

/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:
↪1244:
ConvergenceWarning: Liblinear failed to converge, increase␣
↪the number of
iterations.
  warnings.warn(
```

[6]:
```
model1 = L_SVM (X_train, y_train, max_iter = 4000)
ebaluazioa (X_val, y_val, model1)
```

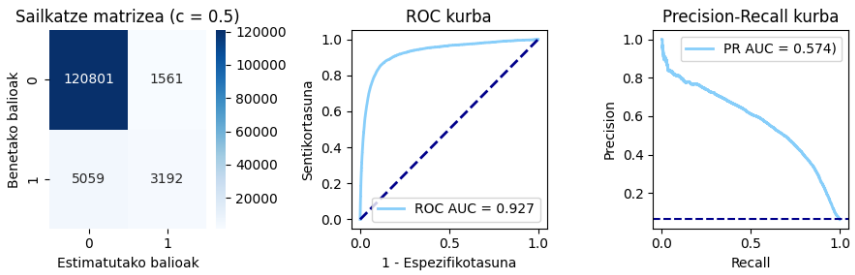
```
PR AUC = 0.5661879934392244
```

[7]:
```
model2 = L_SVM (X_train, y_train, max_iter = 6000)
ebaluazioa (X_val, y_val, model2)
```

```
PR AUC = 0.5661879236878797
```

**Emaitzak**

[8]:
```
ebaluazioa(X_test, y_test, model1, True)
```



```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97    122362
           1       0.67      0.39      0.49      8251

    accuracy                           0.95    130613
   macro avg       0.82      0.69      0.73    130613
weighted avg       0.94      0.95      0.94    130613

accuracy = 0.9493159180173489
PR AUC = 0.5742488826628518
```

## B.7 Decision Tree eta Random Forest

### Datuak kargatu

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
dir = "/content/drive/MyDrive/TFG/"
```

```
Mounted at /content/drive
```

[2]:
```
import pickle
with open(dir + "vdata.bin",mode="rb") as f:
  X_train,X_val,X_test,y_train,y_val,y_test = pickle.load(f)
```

### DECISION TREE EREDUA

#### Eredua sortzen

Decision Tree eredua sortzeko, *scikit-learn*-ek eskaintzen duen *'DecisionTreeClassifier'* funtzioa erabiliko dugu. Honek parametro desberdinak har ditzake, hurrengo link-ean daude ikusgai:

Scikit-learn - DecisionTree

[3]:
```
from sklearn.tree import DecisionTreeClassifier

def decision_tree (X, y, criterion ='gini', max_depth = 20):
  model = DecisionTreeClassifier(criterion = criterion,␣
 ↪max_depth=max_depth)
  model.fit(X, y)
  return model
```

#### Eredua ebaluatzen

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix, roc_curve,␣
 ↪auc, classification_report, precision_recall_curve


def ebaluazioa (X, y, model, full=False):

    y_pred = model.predict(X)
    conf_matrix = confusion_matrix(y, y_pred)
```

```
   # ROC kurba eta AUC-a irudikatzeko:
   y_score = model.predict_proba(X)[:, 1]
   fpr, tpr, _ = roc_curve(y, y_score)
   roc_auc = auc(fpr, tpr)


   # Precisión-Recall kurba irudikatu:
   precision, recall, _ = precision_recall_curve(y, ␣
↪y_score)
   pr_auc= auc(recall,precision)

   if full :
     plt.figure(figsize=(9,3))
     plt.subplot(1, 3, 1)
     sns.heatmap(conf_matrix, annot=True, fmt="d",␣
↪cmap='Blues')
     plt.title('Sailkatze matrizea (c = 0.5)')
     plt.ylabel('Benetako balioak')
     plt.xlabel('Estimatutako balioak')

     plt.subplot(1, 3, 2)
     plt.plot(fpr, tpr, color='lightskyblue', lw=2,␣
↪label='ROC AUC = %0.3f' % roc_auc)
     plt.plot([0, 1], [0, 1], color='darkblue', lw=2,␣
↪linestyle='--')
     plt.xlabel('1 - Espezifikotasuna')
     plt.ylabel('Sentikortasuna')
     plt.title('ROC kurba')
     plt.legend(loc="lower right")

     plt.subplot(1, 3, 3)
     plt.plot(recall, precision, color='lightskyblue',␣
↪lw=2, label='PR AUC = %0.3f)' % pr_auc)
     plt.axhline(y = sum(y) / len(y), color='darkblue',␣
↪linestyle='--')
     plt.xlabel('Recall')
     plt.ylabel('Precision')
     plt.title('Precision-Recall kurba')
     plt.legend(loc="upper right")
     plt.tight_layout()
     plt.show()
```

```
        print('\n', classification_report(y, y_pred))
        print('accuracy =', model.score(X,y))

    print('PR AUC =', pr_auc)
```

**Probak egiten**

Erabaki zuhaitza entrenatzean gini eta entropy erabili ditugu, baina balidazioko emaitzak bien arteko aldeak oso txikiak direla erakutsi dute.

[ ]:
```
model= decision_tree(X_train, y_train, max_depth = None)
ebaluazioa(X_val, y_val, model)
```

```
PR AUC = 0.43145180867507904
```

[ ]:
```
model1= decision_tree(X_train, y_train, criterion=␣
↪'entropy', max_depth = None)
ebaluazioa(X_val, y_val, model1)
```

```
PR AUC = 0.4207893197481743
```

Exekuzioak denbora luzea hartzen zuenez, eta emaitzak oso onak ez zirenez, zuhaitzaren gehiegizko sakonera ezartzea erabaki dugu. Honek, overfiting-a ekiditzen du.

[ ]:
```
model2= decision_tree(X_train, y_train, criterion=␣
↪'gini',max_depth=20 )
ebaluazioa(X_val, y_val, model2)
```

```
PR AUC = 0.40746788260776784
```

[ ]:
```
model3= decision_tree(X_train, y_train, criterion=␣
↪'gini',max_depth=30 )
ebaluazioa(X_val, y_val, model3)
```

```
PR AUC = 0.4254300656656299
```

[ ]:
```
model4= decision_tree(X_train, y_train, criterion=␣
↪'gini',max_depth=50)
ebaluazioa(X_val, y_val, model4)
```

```
PR AUC = 0.4366686723551477
```

```
[ ]: model5= decision_tree(X_train, y_train, criterion=␣
      ↪'gini',max_depth=80)
     ebaluazioa(X_val, y_val, model5)
```

```
PR AUC = 0.4394885083509428
```

```
[ ]: model6= decision_tree(X_train, y_train, criterion=␣
      ↪'gini',max_depth=65)
     ebaluazioa(X_val, y_val, model6)
```

```
PR AUC = 0.4374192528970256
```

```
[ ]: model7= decision_tree(X_train, y_train, criterion=␣
      ↪'gini',max_depth=70)
     ebaluazioa(X_val, y_val, model7)
```

```
PR AUC = 0.4394186516469774
```

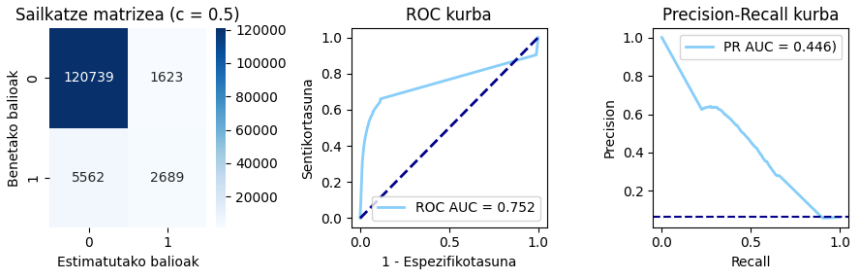
```
[ ]: model8= decision_tree(X_train, y_train, criterion=␣
      ↪'entropy',max_depth=70)
     ebaluazioa(X_val, y_val, model8)
```

```
PR AUC = 0.4367890619875053
```

Ondoriozta dezakegu sakonera maximoa ezartzea ereduaren prozesatze-denbora izugarri murrizten duela. Eta zenbaki desberdinak emanez, max_depth=65 denean ditugu emaitza altuenak. Gini edo entropy erabiltzeari dagokionez, ez dago ia diferentziarik.

**Emaitzak**

```
[ ]: ebaluazioa(X_test, y_test, model7, True)
```



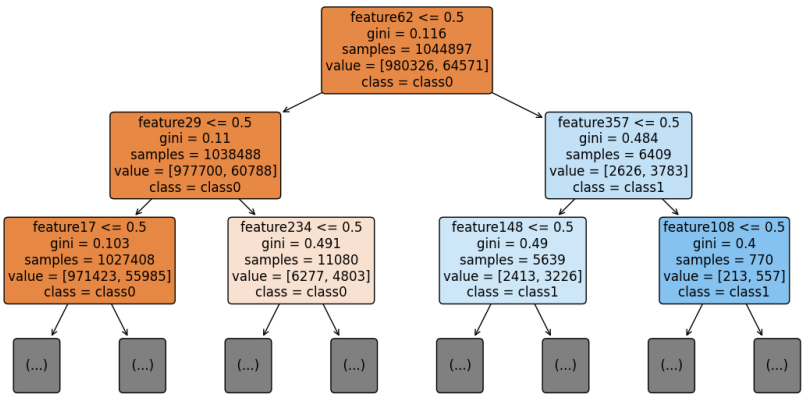
```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97    122362
           1       0.62      0.33      0.43      8251

    accuracy                           0.94    130613
   macro avg       0.79      0.66      0.70    130613
weighted avg       0.93      0.94      0.94    130613

accuracy = 0.9449901617756272
PR AUC = 0.4462155421767052
```

### Decision Tree ereduaren nodoak

Era bisualago batean zuhaitzaren irudikapena ikusteko, lehengo nodoak erakutsiko ditugu. Hauek gini-ren balioa, lagin kopurua eta abar erakutsiko digute.

[ ]:
```
#!pip install graphviz matplotlib
```

[ ]:
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

feature_names = [f'feature{i}' for i in range(X_train.
 ↪shape[1])]
class_names = ['class0', 'class1']
plt.figure(figsize=(14, 7))

# Decision Tree irudikatu:
plot_tree(model7,
          filled=True,
          feature_names=feature_names,
          class_names=class_names,
          rounded=True,
          max_depth=2, #Zenbat nodo
          fontsize=12)

plt.show()
```




## RANDOM FOREST

### Eredua sortzen

RandomForest eredua sortzeko, *scikit-learn* liburutegiak eskaintzen duen *'RandomForestClassifier'* funtzioa erabiliko dugu. Honek parametro desberdinak har ditzake, hurrengo link-ean daude ikusgai:

Scikit-learn - Random Forest

[5]:
```
from sklearn.ensemble import RandomForestClassifier

def random_forest (X, y, criterion = 'gini', max_depth =␣
↪None):
  model = RandomForestClassifier(criterion = criterion,␣
↪max_depth = max_depth)
  model.fit(X, y)
  return model
```

[ ]:
```
#model = random_forest(X_train, y_train ) denbora arazoak
```

### Matrizeen tamaina aldatzen

Eredua entrenatzeko, arazo larri bat izan dugu exekuzio denborarekin. Hau konpontzeko, bektorizatutako matrizearen dimentsioak murriztu behar izan dugu, laginen tamaina gutxituz.

[6]:
```
# n errenkaden kopuru berria
def lagin_tamaina(X_train, y_train,n):

  #print(X_train.shape , X_val.shape, X_test.shape) #␣
↪Hasierako matrizearen tamaina.
```

```
 print(sum(y_train)/len(y_train) * 100) # Mezu toxikoen␣
↪proportzioa.

 X_train = X_train[:n] # Entrenamenduko datuak tamaina␣
↪berrira aldatuko ditugu.
 y_train = y_train[:n]

 print(sum(y_train)/len(y_train) * 100) # Konprobatzeko␣
↪murrizketa egiterakoan datuen klaseen arteko proportzioa␣
↪antzekoa izaten jarraitzen duela.
 #print(X_train.shape , X_val.shape, X_test.shape) #␣
↪Matrize berrien tamaina.

 return X_train, y_train
```

**Probak egiten**

Hasieran, milioi bat mezu inguru genituen, baina 1000 lagineko lagin batekin hasi gara, pixkanaka bikoiztuz eta balidazioko datu multzoekin ebaluatuz.

[ ]:
```
X_train1, y_train1 = lagin_tamaina (X_train, y_train, 1000)

model1 = random_forest  (X_train1, y_train1)
ebaluazioa(X_val, y_val, model1)
```

```
6.179652157102566
5.7
PR AUC = 0.24832712429400125
```

[ ]:
```
X_train2, y_train2 = lagin_tamaina (X_train, y_train, 2000)

model2 = random_forest( X_train2, y_train2)
ebaluazioa ( X_val, y_val, model2)
```

```
6.179652157102566
5.5
PR AUC = 0.2769314929518707
```

[ ]:
```
X_train3, y_train3 = lagin_tamaina(X_train, y_train, 4000)

model3 = random_forest(X_train3, y_train3)
ebaluazioa(X_val, y_val, model3)
```

```
6.179652157102566
6.0249999999999995
PR AUC = 0.3271829779291995
```

[ ]:
```
X_train4, y_train4 = lagin_tamaina(X_train, y_train, 8000)

model4 = random_forest(X_train4, y_train4)
ebaluazioa(X_val, y_val, model4)
```

```
6.179652157102566
6.0249999999999995
PR AUC = 0.38701765158903084
```

[ ]:
```
X_train5, y_train5 = lagin_tamaina(X_train, y_train, 16000)

model5 = random_forest(X_train5, y_train5)
ebaluazioa(X_val, y_val, model5)
```

```
6.179652157102566
6.08125
PR AUC = 0.4138069752262626
```

[ ]:
```
X_train6, y_train6 = lagin_tamaina(X_train, y_train, 32000)

model6 = random_forest(X_train6, y_train6)
ebaluazioa(X_val, y_val, model6)
```

```
6.179652157102566
6.121875
PR AUC = 0.431225944146907
```

[ ]:
```
X_train7, y_train7 = lagin_tamaina(X_train, y_train, 64000)

model7 = random_forest(X_train7, y_train7)
ebaluazioa(X_val, y_val, model7)
```

```
6.179652157102566
6.2078125
PR AUC = 0.4359062472992711
```

[ ]:
```
X_train8, y_train8 = lagin_tamaina(X_train, y_train, 128000)

model8 = random_forest(X_train8, y_train8)
```

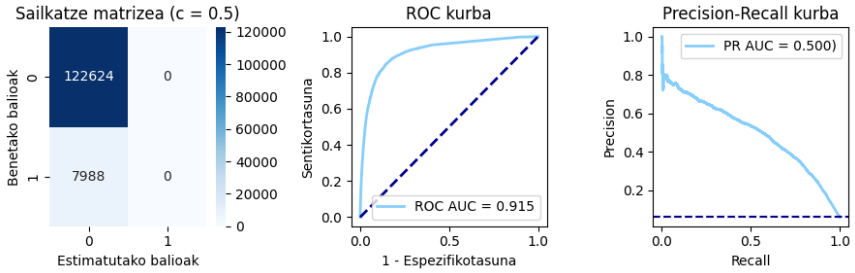
```
ebaluazioa(X_val, y_val, model8)
```

```
6.179652157102566
6.2242187499999995
PR AUC = 0.4500816591905329
```

[7]:
```
X_train9, y_train9 = lagin_tamaina(X_train, y_train, 256000)

model9 = random_forest(X_train9, y_train9)
ebaluazioa(X_val, y_val, model9)
```

```
6.179652157102566
6.20625
PR AUC = 0.4692787898329339
```

[10]:
```
model10 = random_forest(X_train9, y_train9,  max_depth=50)
ebaluazioa(X_val, y_val, model10, True)
```




```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
 ↪_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined␣
 ↪and being set to
0.0 in labels with no predicted samples. Use `zero_division`␣
 ↪parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
 ↪_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined␣
 ↪and being set to
0.0 in labels with no predicted samples. Use `zero_division`␣
 ↪parameter to
```

```
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
 ↪_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined␣
 ↪and being set to
0.0 in labels with no predicted samples. Use `zero_division`␣
 ↪parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

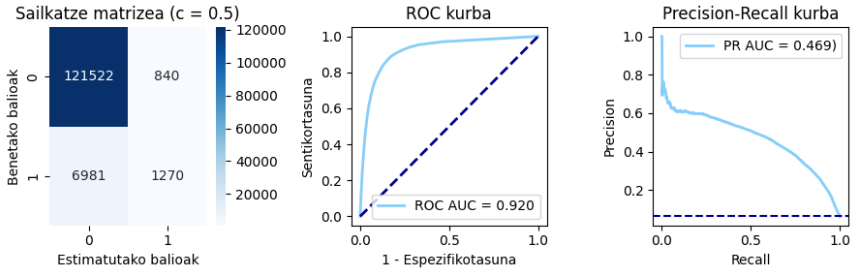
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 1.00 | 0.97 | 122624 |
| 1 | 0.00 | 0.00 | 0.00 | 7988 |
| accuracy | | | 0.94 | 130612 |
| macro avg | 0.47 | 0.50 | 0.48 | 130612 |
| weighted avg | 0.88 | 0.94 | 0.91 | 130612 |

```
accuracy = 0.9388417603283006
PR AUC = 0.5004460077591288
```

Ondoriozta dezakegu, lagin tamaina handitzen den heinean, emaitza hobeagoak lortzen direla. Hala ere, konputazionalki denbora arazoak daudenez, lagin tamaina murriztera behartuta gaude. Eragozpen honek entrenamendu-datuen erdia ere ez hartzera behartzen gaitu gure eredua sortzeko. Azkenik, 256000ko lagin bat erabiltzen dugu.
Eredu honetan ere, parametroak aldatzen sahiatu gara, emaitza optimizatzeko asmoz. "max_depth"murriztean, nahiz eta PR_AUC balio pixka bat handiagoa izan, ereduak ez dituenez positiborik sailkatu, salbuespena egingo dugu. Eredu horrek ez baitu portaera sendoago erakusten sailkapenean eta beste alderdi batzuetan edo metriketan. Hala ere, exekuzio-denbora murrizketa handi bat izan da kontuan izatekoa eta parametroak aldatzeko aukera murriztu duena.

**Emaitzak**

[8]:
```
ebaluazioa(X_test, y_test, model9, True)
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.99 | 0.97 | 122362 |
| 1 | 0.60 | 0.15 | 0.25 | 8251 |
| accuracy | | | 0.94 | 130613 |
| macro avg | 0.77 | 0.57 | 0.61 | 130613 |
| weighted avg | 0.92 | 0.94 | 0.92 | 130613 |

accuracy = 0.9401208149265387
PR AUC = 0.4690298639134738

# Bibliografia


[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. MIT Press, 2018.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[3] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017.

[4] Kaggle, "Kaggle - level up with the largest ai & ml community," https://www.kaggle.com/, 2024, [Online; Accessed: 2024-06-21.

[5] BCAS App. (2022) ¿qué es kaggle y cómo te puede ayudar? Accedido: 2024-06-06. [Online]. Available: https://bcasapp.com/blog/herramientas-tecnologias/kaggle

[6] Kaggle. (2019) Quora insincere questions classification. Accedido: 2024-06-06. [Online]. Available: https://www.kaggle.com/competitions/quora-insincere-questions-classification

[7] W. Blog. (2020) ¿qué es quora y cómo funciona? Accedido: 2024-06-06. [Online]. Available: https://es.wix.com/blog/2020/01/que-es-quora/

[8] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[9] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*,

vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[11] N. L. T. (NLTK). Natural language processing with python. https://www.nltk.org/.

[12] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[13] P. S. Foundation. pickle — python object serialization. https://docs.python.org/3/library/pickle.html.

[14] R. Řehůřek. Introduction to gensim. https://radimrehurek.com/gensim/intro.html.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[16] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[17] M. Mayo, "Preprocesamiento de datos de texto: un tutorial en python," *Datos y ciencia*, vol. 3, 2018.

[18] Tedboy. (sin fecha) gensim.utils.simple_preprocess documentation. [Online]. Available: https://tedboy.github.io/nlps/generated/generated/gensim.utils.simple_preprocess.html#gensim.utils.simple_preprocess

[19] L. Oesper, D. Merico, R. Isserlin, and G. D. Bader, "Wordcloud: a cytoscape plugin to create a visual semantic summary of networks," *Source code for biology and medicine*, vol. 6, no. 1, p. 7, 2011.

[20] T. Shah, "About train, validation and test sets in machine learning," *Towards Data Science*, vol. 6, 2017.

[21] C. E. Metz, "Basic principles of roc analysis," *Seminars in Nuclear Medicine*, vol. 8, no. 4, pp. 283–298, 1978.

[22] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, Apr. 1982.

[23] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, p. e0118432, 2015.

[24] H. He and E. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, Sept 2009.

[25] J. Joyce, "Bayes' Theorem," in *The Stanford Encyclopedia of Philosophy*, Fall 2021 ed., E.N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2021.

[26] D. Berrar, "Bayes' theorem and naive bayes classifier." in *Encyclopedia of Bioinformatics and Computational Biology (1)*. Elsevier, 2019, pp. 403–412.

[27] I. Indriyanto and I. Sumitra, "Measuring the level of plagiarism of thesis using vector space model and cosine similarity methods," in *IOP Conference Series: Materials Science and Engineering*, vol. 662, no. 2. IOP Publishing, 2019, p. 022111.

[28] SciPy, "SciPy - sparse matrices," https://docs.scipy.org/doc/scipy/reference/sparse.html, 2024, [Online; Accessed: 2024-06-21].

[29] D. W. Hosmer and S. Lemeshow, *Applied logistic regression (Wiley Series in probability and statistics)*, 2nd ed. Wiley-Interscience Publication, 2000. [Online]. Available: http://www.amazon.com/Applied-logistic-regression-probability-statistics/dp/0471356328%3FSubscriptionId%3D192BW6DQ43CK9FN0ZGG2%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0471356328

[30] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[31] scikit-learn, "scikit-learn - logistic regression classifier," https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, Consulta: 2024-06-21, [Online; Accessed: 2024-06-21].

[32] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[33] B. E. Boser, I. M. Guyon, and V.N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, p. 144–152.

[34] E. Campo León and J. T. AlcaláNalvaiz, "Introducción a las máquinas de vector soporte (svm) en aprendizaje supervisado," *Trabajo de Fin de Grado en Matemáticas, Universidad de Zaragoza. Obtenido de https://zaguan. unizar. es/record/59156/files/TAZ-TFG-2016-2057. pdf*, 2017.

[35] E. J. C. Suárez, "Tutorial sobre máquinas de vectores soporte (svm)," vol. 1, pp. 1–12, 2014.

[36] scikit-learn, "scikit-learn - linear support vector classifier," https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, Consulta: 2024-06-21, [Online; Accessed: 2024-06-21].

[37] W. Härdle and L. Simar, *Applied Multivariate Statistical Analysis*, fifth edition ed. Cham, Switzerland: Springer, 2019. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2545910

[38] C. J. S. R. O. Leo Breiman, Jerome Friedman, *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[39] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[40] G. Louppe, "Understanding random forests: From theory to practice." Ph.D. dissertation, University of Liège, Belgium, 2014.

[41] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[42] C. Gini, "On the Measure of Concentration with Special Reference to Income and Statistics," *Colorade College Publication*, vol. 208, pp. 73–79, 1936.

[43] S. L. Developers, "Decisiontreeclassifier - scikit-learn documentation," https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, 2024, [Online; Accessed: 2024-06-21].

[44] ——, "Randomforestclassifier - scikit-learn documentation," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2024, [Online; Accessed: 2024-06-21].

[45] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Wiley, May 2000. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471494631

[46] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: http://www.amazon.com/Convex-Optimization-Stephen-Boyd/dp/0521833787%3FSubscriptionId%3D192BW6DQ43CK9FN0ZGG2%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0521833787