Euskal Herriko Unibertsitatea

Erabakiak Hartzeko Euskarri Sistemak

WEKA Text Mining: OneRule eta RandomForest

Egileak:
Paul Castañares, Jose Mari Gonzalez, Jon Ander Asua eta Ander San Juan

https://github.com/JonAnderAsua/WekaProiektua

2021

Gaien Aurkibidea

Irudien aurkibidea

Taulen Zerrenda

1 Sarrera

1.1 Zertan datza testu meatzaritza?

Testu meatzaritza era hetereogeneo batean dauden dokumentuetan (web orrialdeak, mezu elektronikoak,...) informazioa ateratzen du. Hau lortzeko testuan patroiak bilatzen dira, adibidez hitzen maiztasuna edo egitura sintaktikoa.(4)

1.2 Adimen artifiziala eta testu meatzaritza

Hizkuntzaren prozesamenduarekin eta adimen artifizialarekin lotuta dagoen disziplina da. Testu meatzaritzaren helburua, egitura edo formatu gabe dauden testuak dokumentu edo datu baseetara transformatzea da, ondoren datu hauen analisia egiteko, Machine Learning-eko algoritmo bat erabiliz.

1.3 Testu meatzaritzako aplikazioak enpresetan

Zabalduenak dauden aplikazioak honako hauek dira: (1)

- Adimen lehiakorra: enpresek merkatuan duten errendimenduari buruzko ahalik eta informazio hobea lortu nahi dute. Horretarako, posiblea den informazio guztia biltzen dute, baliozkoa dena ateratzen dute eta multzo ezberdinetan banatzen dute, gero multzoka analisia egiteko.
- Giza baliabideak: langileen iritziak aztertzeko eta langile berrien aukeraketarekin laguntzeko, curriculum-en analisia burutuz.
- Bezeroarekiko erlazioaren kudeaketa: bezeroen mezuen edukia aztertzen da, gehien bat sarritan egiten diren galderei erantzuna zuzenean emateko.
- Merkatuaren analisia: testu meatzaritzak merkatu egoera konplexuagoei lotuta dauden kontsultei erantzuna emateko gai izateko aukera ematen du.

1.4 Testu meatzaritzak dakartzan erronkak datu numerikoekin alderatuta

Testu meatzaritzak, datu numerikoek ekartzen ez dituzten erronkak sortzen ditu:

- Jasotako testuak edozein motakoak izan daitezke (.csv, .txt, .odt...), beraz, meatzaritzan erabiliko diren tresnetan baliagarriak diren formatuetara konbertitzea beharrezkoa da (gure kasuan, WEKA ingurune grafikoan, .arff fitxategiak behar ditugu).
- Testuak idazterako orduan askatasuna handiagoa da, beraz, jaso daitezkeen datuen eremua zenbakizko datuena baino handiagoa dela esan dezakegu.
- Testuetan karaktere bereziak egon daitezke, hainbat filtro aplikatzea oztopatzen dituztenak.
- Hitzak eta esaldiak dituzten esanahiak ironikoak, retorikoak.. izan daitezke.
- Zenbakiak taldekatzeko, formulak aplikatzeko... errazak dira, testuetan, berriz, edozein gauza egon daiteke ezin daitekeena bide berdinetik ebaluatu.

2 Aurre-prozesamendua

2.1 Datu errepresentazioen ezaugarriak

- Bag of Words (BoW): (2)
  - Instantziak bektore binarioa bihurtu non atributu bakoitza hitz hori agertzen den ala ez den.
  - Ez du esaldiaren ordena adierazten.
  - Hitz bat errepikatuta dagoen ala ez ezin da jakin.
- Raw: Testu xehea, ez du irudurik, bakarrik hitzak.
- Term Frecuency - Inverse Document Frecuency (TF-IDF): (5)
  - Instantziak osatzen duten hitzen maiztasunean oinarritu.
  - Maiztasun txikiko hitzetan bideratu.
  - Hainbat testu ezberdinekin lan egitea ahalbidetu.
  - Hitzaren nabarmentasuna aztertu.

2.2 Transformazio bakoitzaren adibideak

Adibide bezala hurrengo esaldi multzoa (korpusa) erabiliko dugu:

It was the best of times
It was the worst of times
It was the age of wisdom
It was the age of foolishness

- Raw: Kasu honetan gure korpusean ez da ezer aldatzen. Aurreko puntuan azaldu dugun bezala "Raw"formatua testu xehea da.
- Bag of Words: (2)
  - Esaldi hauetan agertzen diren hitzak zerrendatuko dira, hauek gure .arff fitxategiaren atributuak izango dira.

    it
    was
    the
    best
    of
    times
    worst
    age
    wisdom

- Lehenengo puntuko esaldiak ateratako atributuekin bektore binario bihurtu.
  It was the best of times = [1,1,1,1,1,1,0,0,0,0]
  It was the worst of times = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
  It was the age of wisdom = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
  It was the age of foolishness = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
- Bektore hau '0'z beteta egon ahal denez 'sparse' egingo da erabilgarria ez den informazioa kentzeko.
  It was the best of times = 1 1,2 1,3 1, 4 1,5 1,6 1
  It was the worst of times = 1 1,2 1,3 1,5 1,6 1,7 1
  It was the age of wisdom = 1 1,2 1,3 1,5 1,8 1,9 1
  It was the age of foolishness = 1 1,2 1,3 1,5 1,8 1,10 1

- TF-IDF: (6) (11)
  Hurrengo formula erabiliko dugu "times" hitzaren pisua kalkulatzeko:

$$Wt, d = TFt, dlog(N/DFt) \tag{1}$$

  Non:
  log (N/DFt) = IDFt
  TFt,d: "d" dokumentuan "t" terminoaren agerpen kopurua
  DFt: "t" terminoa duten dokumentu kopurua
  N: Aztertu nahi ditugun dokumentu guztiak

  Kalkuluak:
  IDFtimes = log(4/2) = 0.301
  TFtimes = 2/24 = 0.083
  Wt = 0.083 * 0.301 = 0.025

  TF-rekin hitzaren pisu osoa lortzen dugu korpus (dokumentu sorta) osoan eta IDF-rekin biderkatzerakoan pisu horren inbertsioa egiten dugu.

2.3 Softwarearen xehetasun nabarienak

- Lehenengo atala:
  - GetRaw: "Raw" formatoan dagoen fitxategia datu multzoz osatutako .arff fitxategi bat bihurtzen du. Horretarako R lengoaian idatzitako script bat sortu dugu.
    Pakete hau exekutatzeko hurrengo komandoa erabiliko da:
    ./getRaw.sh train.txt dev.txt testBlind.txt weka.jar
  - MakeCompatible:Klasea eta String motatako atributu bat duen .arff fitxategi bat bektore sorta bat bihurtzeko erabiliko da. Horretarako Bag of Words (BoW) eta Term Frecuency - Inverse Document Frecuency (TF-IDF) erak erabiltzeko aukera eman da. Ateratako bektorea Sparse ala Non-Sparse erara egon daiteke ere.
    Pakete hau exekutatzeko hurrengo komandoa erabiliko da:
    java -jar TransformRaw.jar train.arff hiztegia IDFTF(YES/NO) TFTF(YES/NO) SPARSE(YES/NO)
  - TransformRaw: Entrenamendu eta ebaluazio multzoen atributuak ezberdinak izan ahal direnez azkenaren atributu multzoa egokitu behar da, horretarako StringtoWordVector filtroa entrenamendu multzoan aplikatu ostean ateratako hiztegiarekin filtro bat sortuko dugu eta ebaluazio multzoa handik pasatuko dugu.
    Paketea exekutatzeko hurrengo komandoa erabiliko da:
    java -jar MakeCompatible.jar dev.arff hiztegia spamDevBoW
- Bigarren atala:
  - ParametroEkorketa:
  - GetOneRModel: Baseline modeloa, gure kasuan One Rule, sortu, train instantzia multzoarekin entrenatu ebaluazio ez zintzoa, Cross Validation eta Hold Out erabiliz eta bukatzeko entrenatutako modelo hori .model fitxategi batean gorde.
    Pakete hau erabiltzeko hurrengo komandoa erabiliko da:
    java -jar GetOneRModel train.arff modela.model kalitatea.txt
  - GetRandomForestModel: Esleitutako modelo konplexua, gure kasuan random Forest, sortu, train instantzia multzoarekin entrenatu ebaluazio ez zintzoa, Cross Validation eta Hold Out erabiliz eta bukatzeko entrenatutako modelo hori .model fitxategi batean gorde.
    Pakete hau erabiltzeko hurrengo komandoa erabiliko da:
    java -jar GetRandomForestModel train.arff modela.model kalitatea.txt
  - AtributuHautapena: Entrenamendu instantzia multzoan AttributeSelection filtroa erabiliko da bakarrik klasearekin erlazio handia eta beste atributuekin erlazio txikia duten atributuekin geratzeko. Honetaz aparte ebaluazio multzoa egokituko da entrenamendu multzoan kendutako atributu berdinak kentzeko.

Pakete hau erabiltzeko hurrengo komandoa erabiliko da:
java -jar AtributuHautapena train.arff hiztegia.txt test.arff

- Hirugarren atala:
  - Predictions: Esaldi bat edo instantzia multzo bat (.arff formatuan) eta modeloa sartuta .txt formatuan dagoen fitxategi bat bueltatzen du sartutakoaren iragarritako klasearekin edo klaseekin. Horretarako sartutakoa aurretik ateratako hiztergiarekin filtratzen da eta ondoren ebaluatzen da Predictionsekin iragarritako klasea jakiteko.
    Pakete hau erabiltzeko hurrengo komandoa erabiliko da:
    java -jar Predictions modeloa.model (.arff fitxategia edo esaldia) iragarpenak.txt

2.4 Atributuen Hautapena - Azalpena

Atributuen hautapen teknikak erredundanteak ala garrantzi gutxiko atributuak baztertzeko erabiltzen dira. Helburua, datuetan dauden atributuak aztertu eta informazio irabazia maximizatzen dutenak mantentzea da, gure atributu espazioaren dimentsioa txikitzeko. Honi esker, instantzien errepresentazioa sinplifikatu eta sailkapen prozesua azkartuko da. Teknika hauek ondo erabiltzea ezinbestekoa da, sinplifikatzeak informazioa galtzea ekar dezake eta.
AttributeSelection klasea erabili daiteke atributuen hautapena egiteko. Ondoren, klase honetan InfoGainAttributeEval ebaluatzailea eta Ranker bilatzailea finkatzen dira. Azkenik, filtro hau instantziei aplikatuz, bilatzailean zehaztutako aukeren arabera, informazio irabazi gehien duten atributuak mantendu eta gainontzekoak baztertu egingo dira. (7)

3 Esparru Teorikoa

3.1 Baseline - One Rule

Konfigurazio aldetik eta inferentzia denbora aldetik sinplea den eredua da Baseline algoritmoa. Eredu konplexu bat sortzeko, ezinbestekoa da hasieran Baseline bat sortzea, honek adieraziko baitu izan beharreko kalitatearen behe bornea. Eredu honen bitartez lortuko dugu gure eredu konplexuak izan beharreko kalitate minimoa jakitea, modelo biak konparatzen. Gure eredu konplexua oinarrizko ereduaren kalitatea gainditzen ez badu, ez da zentzuzkoa hau erabiltzea, ez baititu hobekuntza esanguratsuak ekarriko. Gure kasuan One Rule sailkatzailea izan da oinarria.

One Rule sailkapen algoritmo sinplea eta merkea da, arau sorta bat sortzen duena iragarle bakoitzeko. Errore minimoa duen atributua erabiltzen du, klasearekiko korrelazio handiena duena, eta honen bitartez finkatzen da instantzia bakoitzaren klasea. Maila bakarreko erabaki zuhaitza sortzen du, atributu zehatz bat frogatzen duten arau multzo batekin adierazten dena. Arau multzo hauek zehaztasun handia lortzen dute gehienetan. (10) (8)

OneR-ren funtzionamendua ulertzeko algoritmoaren pseudokodea erabiliko da:

```
Algorithm: OneRule(Ztrain)
        require: Ztrain
        where:
                C = {c1,c2,...,cn} Klaseen multzoa
                Vi = {vi1,vi2,...,vin} Atributuen balio multzoa
                X = {x1,x2,...,xn} Atributu bakoitzaren balio multzoa
        ensure: One Rule modeloa

        begin:
                for(xi ∈ X)
                        for(ck ∈ C)
                                for(vij ∈ Vi)
                                        N(vij) = N[z ∈ Ztrain: xi = vij ^ C = ck]
                                        K(ck) = N
                for(xi ∈ X)
                        e = Σ[argmin K(ck)(vij)]
                        E(xi) = e / N(Ztrain)
                return xi ∈ X / argmin E(xi)
        end
```

Irudia 1: One Rule algoritmoa

3.2 Esleitutako Algoritmoa - Random Forest

Random Forest aurkezteko lehenengo erabaki zuhaitz bat zer den azaldu behar da. Erabaki zuhaitz edo decision tree batek lotutako hainbat erabakien balizko emaitzen mapa da, haren akzioak kostua, probabilitatea eta irabazien arabera konparatu ditzakeena.

Erabaki zuhaitz batek root izeneko nodo bakar batetik hasten da normalean eta handik emaitza posibleen artean adarkatzen da. Emaitza hauek sortutako adar bakoitzak nodo bat sortzen du eta prozesua berriro egiten da baina orain "root" nodoa hartu beharrean adarraren nodoa hartzen da erreferentzia bezala, hau zuhaitz era ematen dio. Hiru nodo mota daude:

- Probabilitatezko nodoa: Emaitza baten probabilitatea ematen du. Zirkunferentzia batekin adierazten da.
- Erabaki nodoa: Zein erabaki hartzen den adierazten du eta lauki batekin adierazten da.
- Bukaerako nodoa edo hostoa: Emaitza jakin batzuk hartzearen ondorioz ateratako emaitza finala, hau da, "bidearen bukaera". Triangelu batekin adierazten da.

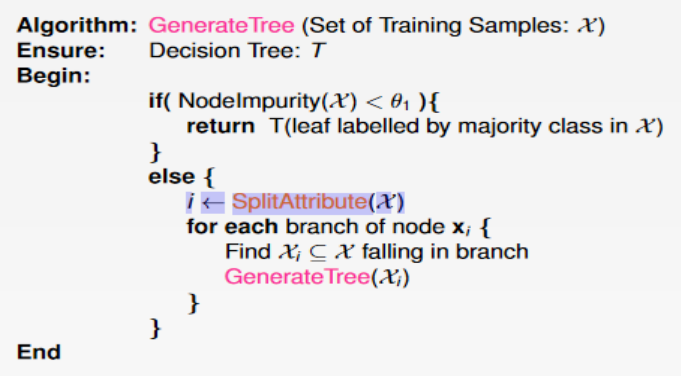
Erabaki zuhaitz bat hurrengo kasuetan erabitzea komeni da:

- Ekintza edo ibilbide aukerak ondo zehaztuta daudenean. Adibidez, proposamen bat onartzea edo baztertzea, ekoizpen-gaitasuna handitzea edo ez, ...
- Ziurgabetasunak kuantifikatu daitezkenean. Adibidez, publizitate-kanpaina batek arrakasta izateko probabilitatea, salmentetan eragina izateko probabilitatea, etapak gainditzeko probabilitatea, ...
- Helburuak zehatzak direnean. Adibidez, Salmentak handitzea, erabilgarritasunak maximizatzea, kostuak minimizatzea, ...

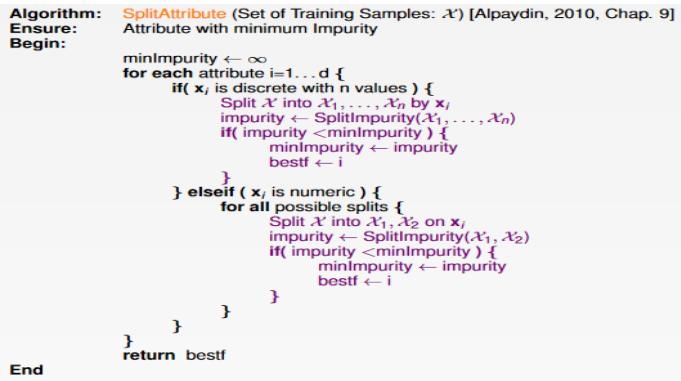
Random forest batek hainbat erabaki zuhaitzak hartzen ditu batera lan egiteko. Erabaki zuhaitz bakoitzak haren iragarpenak egiten ditu eta random forest-ek iragarpen bezala zuhaitz guztiek iragarritako klase maioritarioa adierazten du. Gakoa dator ikustean random forest modelo bezala erabilitako decision tree guztiak erlazio gutxi daukatela, horren ondorioz, elementu bakar batek baino zehaztasun handiagoa lortzen da zuhaitzek haien artean babesten direlako erroreen artean. Zuhaitz batek instantzia bat txarto iragatzen duenean beste asko ondo iragarriko dute instantzia bera, horri esker bide egoki bat jarraitu ahal da. Random forest batek bi baldintza bete behar ditu: (13)

- Beharrezkoa da funtzioetan benetako seinaleren bat egotea funtzio horiekin sortutako ereduek ausazko igarpenak baino hobeto funtziona dezaten.
- Zuhaitz bakoitzak egindako predikzioak erlazio oso txikia izan behar dute besteekik konparatuta.

Random Forest eta Decision Tree baten algoritmoak hurrengoak dira:



Irudia 2: GenerateTree algoritmoa



Irudia 3: SplittAttribute algoritmoa

Algorithm 1: Pseudo code for the random forest algorithm

To generate c classifiers:
for i = 1 to c **do**
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N_i containing D_i
  Call BuildTree(N_i)
end for

BuildTree(N):
if N contains instances of only one class **then**
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create f child nodes of N, N_1,..., N_f, where F has f possible values (F_1, … , F_f)
  for i = 1 to f **do**
    Set the contents of N_i to D_i, where D_i is all instances in N that match F_i
    Call BuildTree(N_i)
  end for
end if

Irudia 4: Random Forest algoritmoa

4 Esparru Esperimentala

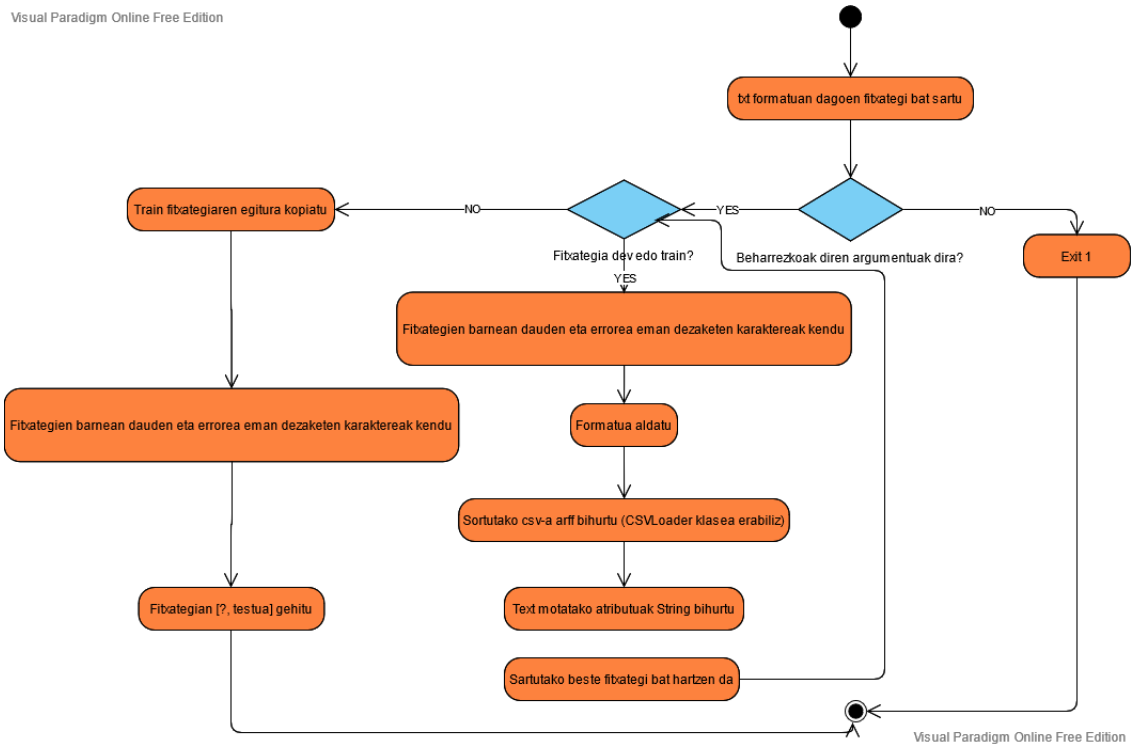
Esparru esperimentala hasi baino lehen aipatzekoa da frogak egiteko hurrengo mikroprozesadorea erabili dela: Intel i5 - 10400 4.3 GHz

4.1 Software pakete bakoitzaren diagrama

Hurrengo irudietan software bakoitzaren funtzionamendua fluxu diagrama simple baten bidez adieraziko dira.

4.1.1 GetRaw

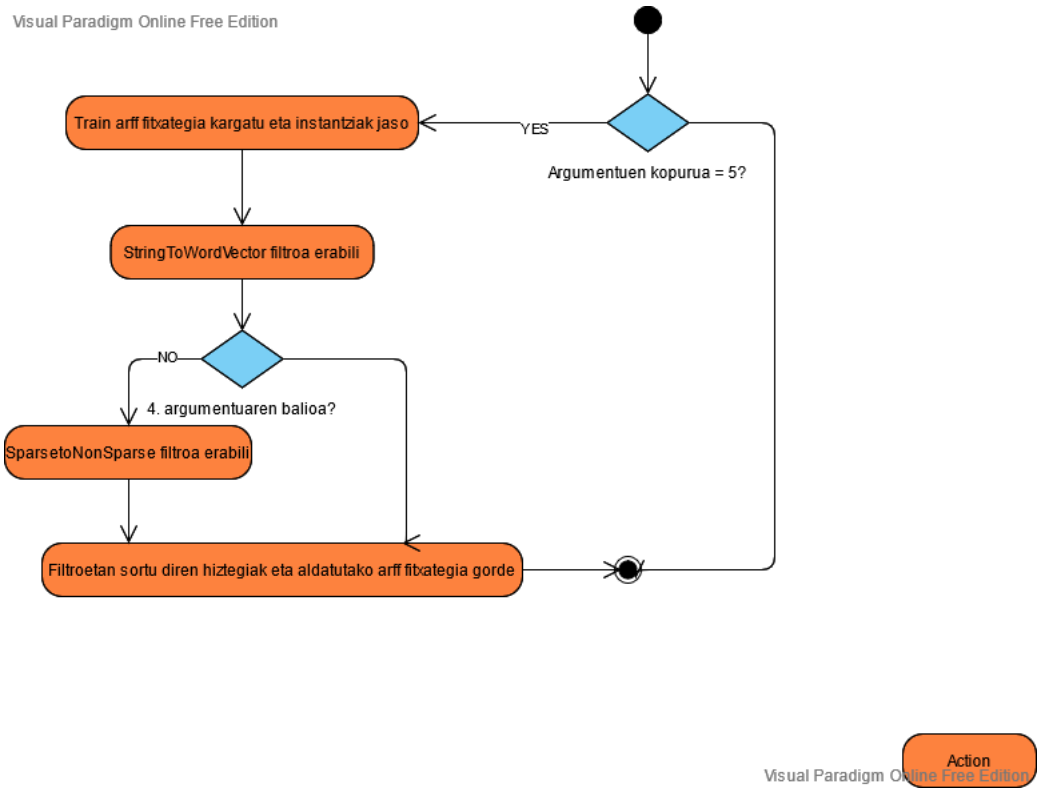
Train.txt, Dev.txt, Test.txt eta weka.jar fitxategien helbidea sarturik, hiru fitxategiak .arff formatura bihurtuko ditu programa honek



Irudia 5: GetRaw programaren fluxu diagrama

4.1.2 TransformRaw

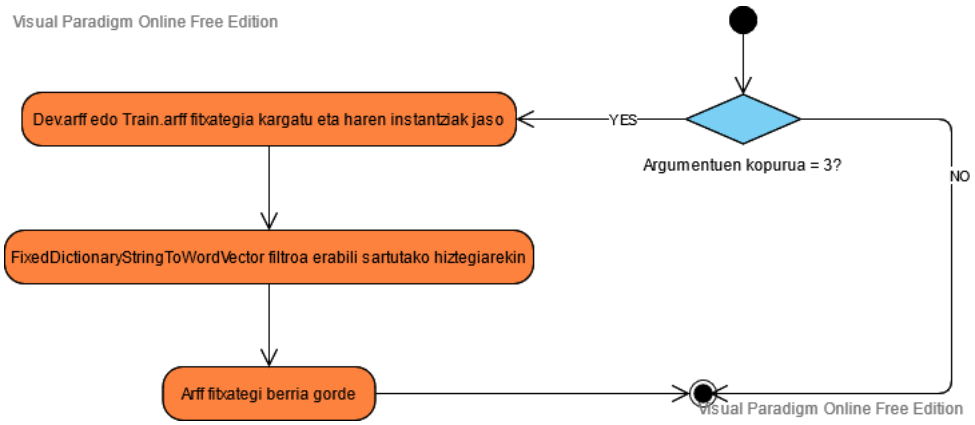
GetRaw programatik lortutako .arff fitxategia, gordeko den hiztegiaren helbidea, IDF transformazioa nahi den ala ez, TFT transformazioa nahi den ala ez eta Sparse edo NonSparse izatea nahi den ala ez sarturik, .arff fitxategi berri bat sortuko du.



Irudia 6: TransformRaw programaren fluxu diagrama

4.1.3 MakeCompatible

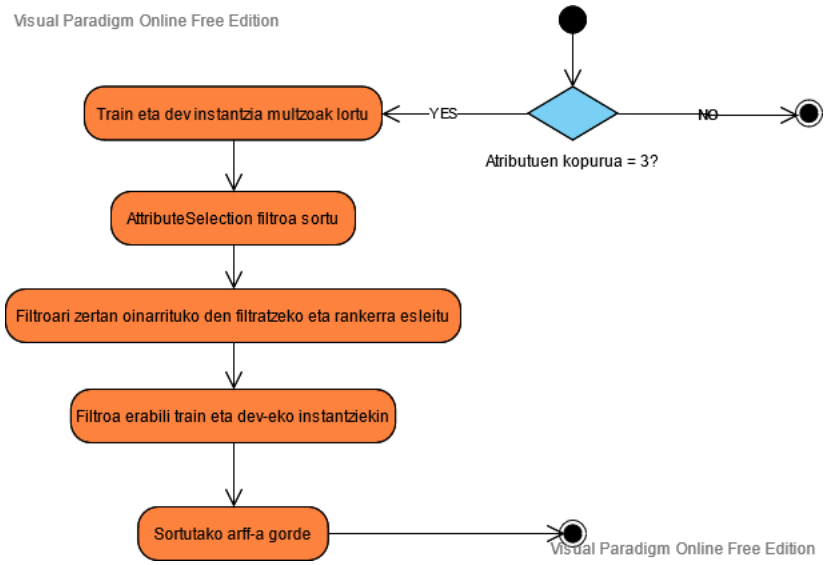
Dev edo test .arff fitxategia, hiztegiaren helbidea eta irteerako .arff fitxategiaren helbidea sarturik train fitxategiarekin bateragarria den .arff fitxategi bat sortuko.



Irudia 7: MakeCompatible programaren fluxu diagrama

4.1.4 AtributuHautapena

Train.arff fitxategia, hiztegi berria gordetzeko helbidea eta dev ala test .arff fitxategiak, hasieran lortutakoak, ez MakeCompatible-n lortutakoak, sarturik, atributu hautapena egingo du, hiztegi berria gorde eta train fitxategiarekin bateragarria den .arff fitxategi bat sortuko.

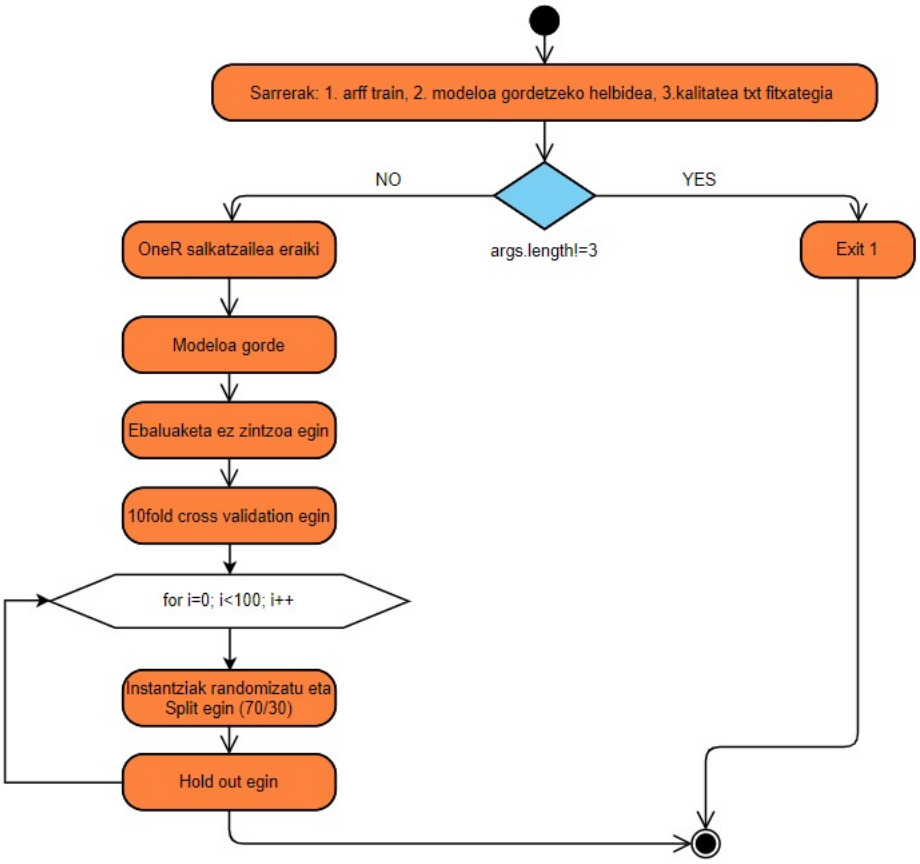


Irudia 8: AtributuHautapena programaren fluxu diagrama

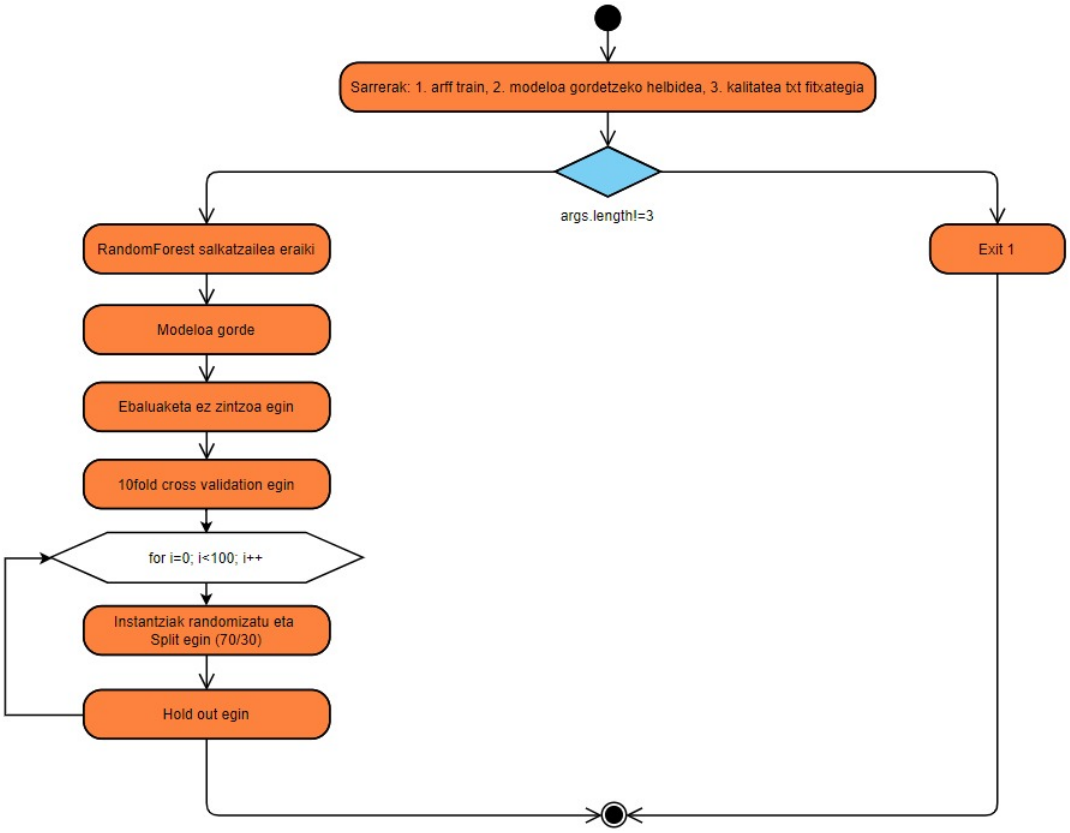
4.1.5 GetOneRModel eta GetRandomForestModel

Train multzorako erabiliko den .arff fitxategia, modeloa gordetzeko erabiliko den helbidea eta emaitzak gordetzeko erabiliko den fitxategia sarturik, aukeratutako modeloa sortuko du eta hiru ebaluazio egingo ditu:

1. Ebaluazio ez zintzoa
2. 10-fold cross validation train multzoarekin.
3. 70/30 hold-out train multzoarekin.



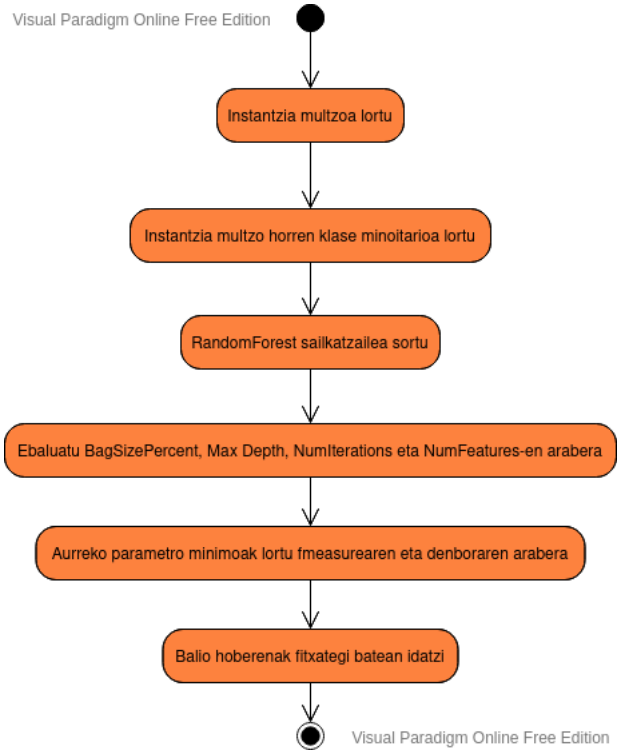
Irudia 9: OneRule programaren fluxu diagrama



Irudia 10: RandomForest programaren fluxu diagrama

4.1.6 ParametroEkorketa

Parametro ekorketa egiteko erabili nahi den instantzia multzoa eta emaitzak gordetzeko erabili nahi den fitxategia sarturik, parametro ekorketa egikarituko du eta parametroen balio optimoa kalkulatuko du klase minoritarioaren FMeasure erabiliz.



Irudia 11: ParametroEkorketa programaren fluxu diagrama

4.1.7 Predictions

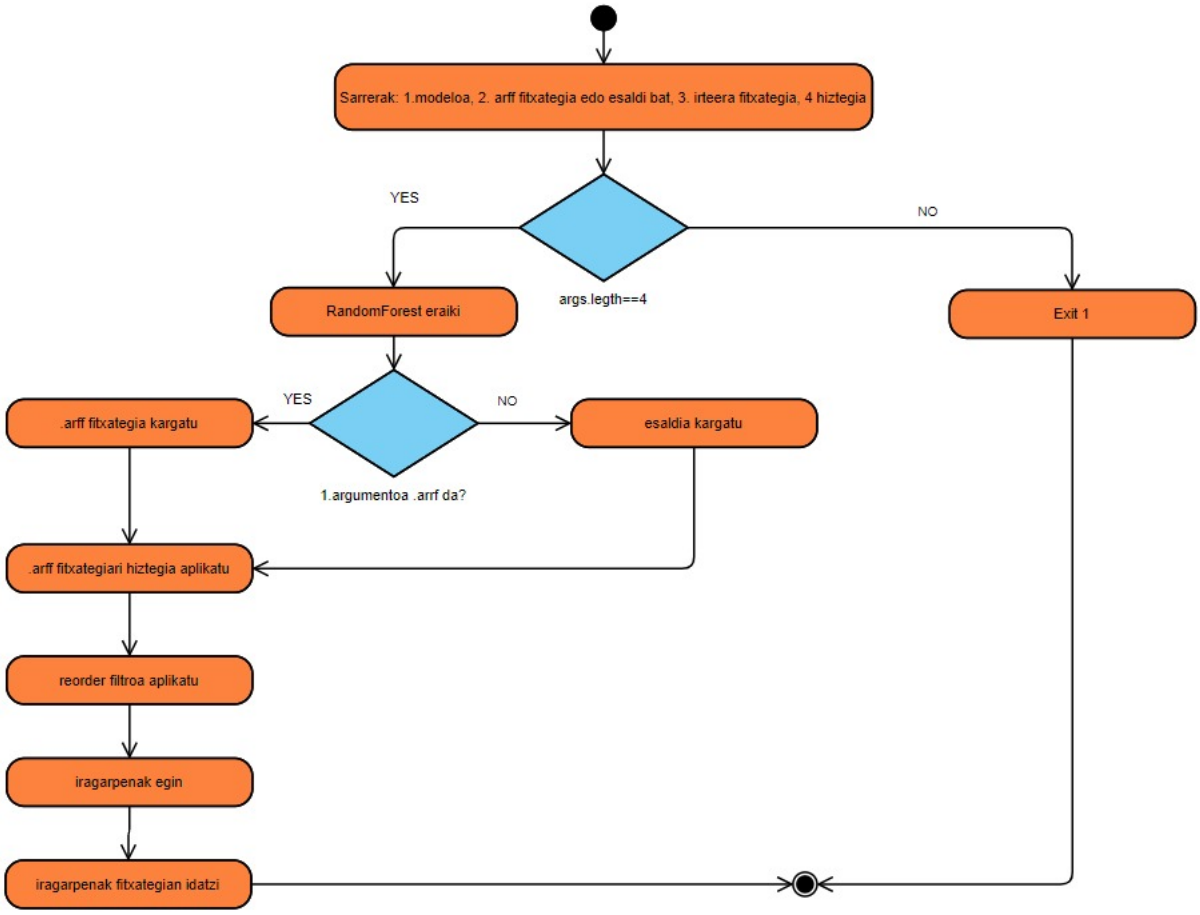
Programa honek bi aukera ditu:

- .arff fitxategi bateko instantziak sailkatu (test.arff adibidez):

  Hiru argumetu hartuko ditu programak, erabili nahi den modeloa, .arff fitxategia eta emaitzak gordetzeko erabiliko den fitxategia.

- Esaldi bat sailkatu:

  Hiru argumentu hartuko ditu, erabili nahi den modeloa, sailkatu nahi den esaldia (komatxoen artean) eta emaitza gordetzeko erabiliko den fitxategia.



Irudia 12: Predictions programaren fluxu diagrama

4.2 Emaitza nabarienak

Emaitza nabarienak aztertzeko ParametroEkorketa software paketean ateratako FMeasure eta denbora (segundutan) ezberdinetan oinarrituko gara. Lortutako balio optimoak hurrengo taulan adieraziko dira (Ikusi Taula 1):

Parametroa	Puntu Optimoa
bagSizePercent	16
MaxDepth	50
numFeatures	200
numIterations	26

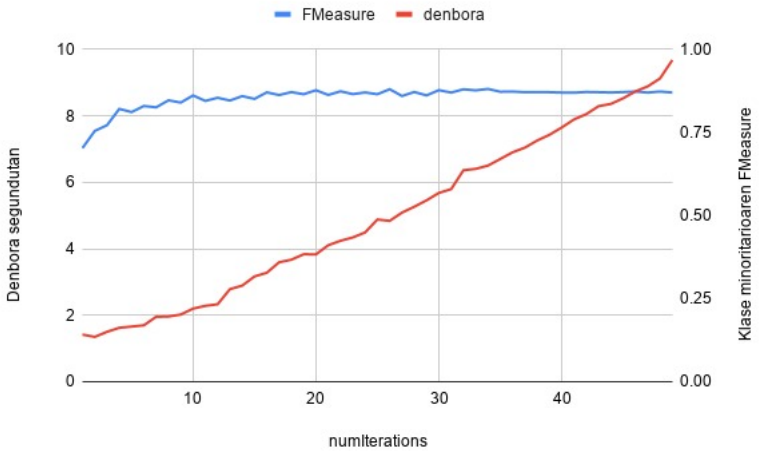
Taula 1: Balio optimoen taula

Grafikoak aztertu baino lehen pare bat kontzeptu azaldu behar dira:

- Bootstrap agregazioa: n tamainako entrenamendu multzo bat emanda m entrenamendu multzo sortzen dira n' tamainakoak. Sortutako entrenamendu multzo horiek batzen dira sailkatzaile orokor sortzeko. (12)
- bagSizePercentage: Aurreko puntuan (Bootstrap agregazioa) sortutako entrenamendu multzoak jatorrizko entrenamendu multzoaren instantziaren ehunekoa.
- numIterations: Basoan egongo den poltsa (zuhaitz) kopurua. (3)
- numFeatures: Zatiketa bakoitzean kontuan hartuko diren atributu kopurua. (3)

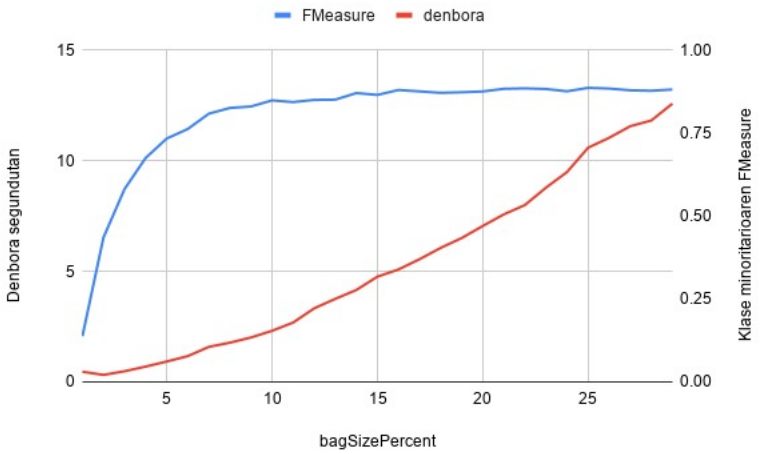
Ondoren parametro bakoitzaren ekorketa egin dugu, beste hirurak lortutako balio optimoan mantenduz, parametro bakoitzaren pisua eta erabilgarritasuna ikusteko.

- Iterazio kopurua: Lehenengo grafiko honetan ikusten da nola iterazio kopurua aurrera doan einean FMeasure eta denbora aurrera doaz ere eta 20. iterazioan kalitatea neurtzeko erabili dugun parametroak balio ia konstante bat hartzen duela. (ikusi 13. irudia)



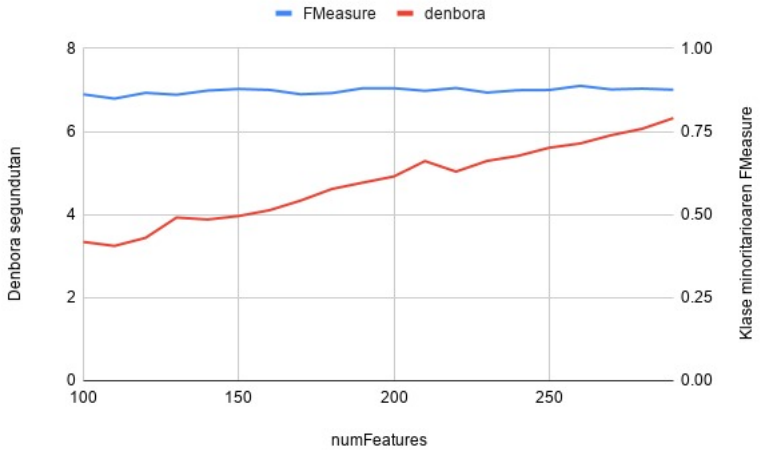
Irudia 13: Iterazioak - Denbora grafikoak

- Bootstrap agregazioa: Hurrengo grafiko honetan ikus daiteke nola bagSizePercentage-aren balioa igotzen denean denbora eta FMeasure igotzen da baina 13. irudiaren grafikoa bezala FMeasure-a puntu batean balio ia konstante bat hartzen du. Kasu honetan balio konstante hori bagSizePercentage = 15 denean gertatzen da. (Ikusi 14. irudia)



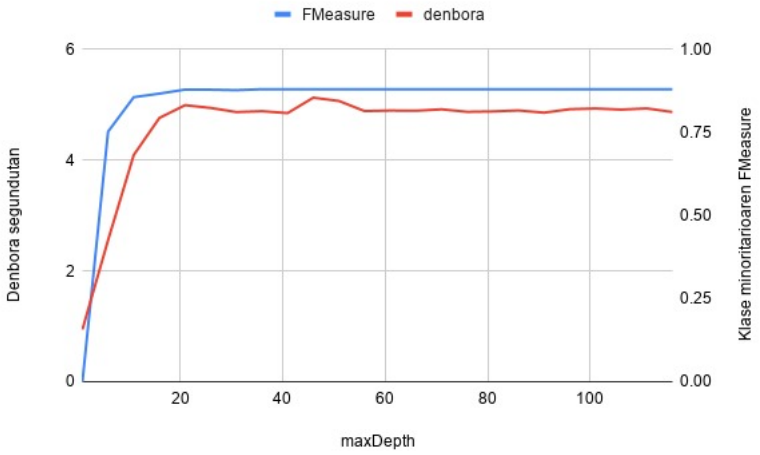
Irudia 14: bagSizePercentaje - Denbora grafikoak

- Ezaugarri kopurua: Hirugarren grafiko honetan FMeasure eta denbora ezaugarri kopuruen (numFeatures) arabera adierazten da. 15. irudian ikusten denez FMeasure balio ia konstantea du momentu oro, ondorioz esan dezakegu aztertutako parametro hau ez daukala garrantzi handirik parametro ekorketa egiterako momentuan.



Irudia 15: numFeatures - Denbora grafikoak

- Sakonera maximoa: Azkenengo grafiko honetan ikus daiteke nola FMeasure eta denbora igarotzen diren hartutako sakonera maximoaren arabera. 16. irudian adierazten den bezala, bertikalean dauden parametroak igoera handi bat dute hasieran baina 40. puntuaren inguruan haren balioak ia konstante bihurtzen dira.



Irudia 16: maxDepth - Denbora grafikoak

4.3 BoW ala TF-IDF

Hasteko BoW eta TF-IDF erabiltzearen ondorioz ateratako FMeasure-ak eta zenbat denbora behar izan duten hartuko dira. Froga hau egiteko erabili den sailkatzailea RandomForest da eta datuei atributu hautapena egin zaie. Hurrengo taulan argitaratuko dira emaitzak: (Ikusi Taula 2)

-	FMeasure	Denbora(s)
BoW	0.8803	8.469
TF-IDF	0.8389	8.7217

Taula 2: BoW eta TF-IDF Kalitate eta denbora balioen taula

Bag of Words datu errepresentazioa FMeasure handiagoa dauka eta honetaz aparte denbora gutxiago behar du, ondorioz esan dezakegu Bag of Words erabiltzea onuragarria dela kasu honetan.

4.4 Atributuen hautapena

Atributuen hautapena egitea egokia den ala ez jakiteko aurreko puntuetan erabili ditugun parametroak erabiliko dugu, hau da, FMeasure eta denbora (segundutan). Frogak egiteko RandomForest sailkatzailea eta Bag of Words datu errepresentazioa erabili dira. Hurrengo taulan parametro horien balioak argitaratuko dira: (Ikusi Taula 3)

Atributu Hautapena	FMeasure	Denbora(s)
Ez	0.8069	8.5834
Bai	0.8803	8.4698

Taula 3: Atributu hautapenaren Kalitate eta denbora balioen taula

Datuak konparatzen hasi aurretik esatekoa da mikroprozesagailuaren hari guztiak erabiltzeko RandomForest sailkatzailea eskaintzen duen setNumExecutionSlots metodoa erabili da. Metodo honek mikroprozesagailuaren zenbat hari erabili erabili nahi diren esleitzen ditu. Honi esker denbora murriztea lortu da, hasieran 50 segundu baino gehiago behar zuen eta orain bakarrik taulan agertzen den balioa.(9) Hau azalduta datuak aztertuko dira.
Atributu hautapena egiteak FMeasure handiagoa ematen du eta honetaz aparte denbora gutxiago behar du, ondorioz esan dezakegu fitxategiari atributu hautapena egitea onuragarria dela kasu honetan.

4.5 Eredua

Baseline edo esleitutako eredua (RandomForest) erabiltzeko aurreko puntuetan erabili ditugun parametroak erabiliko dugu, hau da, FMeasure eta denbora (segundutan). Frogak egiteko atributu hautapena egin da eta Bag of Words datu errepresentazioa erabili daa. Hurrengo taulan parametro horien balioak argitaratuko dira: (Ikusi Taula 4)

Sailkatzailea	FMeasure	Denbora(s)
OneRule	0.2684	6.0883
RandomForest	0.8803	8.4698

Taula 4: Sailkatzaileen Kalitate eta denbora balioen taula

Nahiz eta RandomForest sailkatzaileak denbora gehiago erabiltzen duen haren FMeasure-a nahiko handiagoa da OneRule-ek daukana baino, gainera denboren desberdintasuna ez da hain handia, ondorioz esan dezakegu RandomForest sailkatzailea erabiltzea oso egokia dela kasu honetan.

4.6 Diskusioa

Azterketa sakon bat egin eta gero, gure RandomForest eredua oso zehatza dela ikusi dugu, izan ere 97.8737-an sailkapenak ondo egiten ditu. Hori dela eta, eredua bakarrik erabilita erantzun nahiko zuzenak lortzen direla aurreikusi dezakegu. Ala ere, aipatu beharreko da eredu hau ez dela oso azkarra, datu sorta asko aztertzea beharrezko duten eginkizunak denbora asko eman dezakete erantzunak ematen.
Bestalde OneRule eredua askoz azkarragoa da, eta nahiz eta RandomForest baino errore gehiago euki, oraindik ere nahiko zehatza dela ezan dezakegu, instantzien %88.7211 ondo sailkatzen baitditu. Gure ustez egindako esperimentuak eredua ebaluatzeko nahikoak dira, ala ere, esperimentu gehigarriak egin daitezke erantzun horiek ziurtatzeko eta egindako errore posibleak baztertzeko.
Amaitzeko, esan dezakegu, proiektu hau testu meatzaritzako antzeko atazetarako erabilgarria izango litzakeela. Azken finean, testu meatzaritzako edozein datu aztertzeko egitura berdina erabiliko delako, beraz, bateragarria izango litzateke.

5 Ondorioak

Lan honetan SMS mezuen sailkapena egin da, testu jakin bat emanda testu hori SPAM den ala ez iragarriko duena.
Bukatzeko, esparru esperimentala kontuan izanda hurrengo ondoriotera ailegatu gara:

- RandomForest sailkatzailea erabiltzea hobeagoa da. 4.5 atalean azaltzen den bezala esleitutako algoritmoak baseline baino emaitza hobeagoak ematen ditu, ondorioz bezeroari sailkatzaile horrekin eraikiko zaio modeloa.
- 4.2 puntuan azaltzen den bezala lau dira lan honetako parametro esanguratsuenak:
  - numIterations: Egikaritutako iterazio kopurua.
  - bagSizePercent: 4.2 puntuaren hasieran azalduta dago.
  - numFeatures: Erabilitako ezaugarri kopurua.
  - maxDepth: Erabilitako sakonera maximoa.
- 4.3 atalean Bag of Words (BoW) eta Term Frecuency - Inverse Document Frecuency (TF-IDF) aztertu ondoren hauek emandako emaitzak ikusita Bag of Words errepresentazioa erabiltzea erabaki da puntu horretan azaltzen den bezala haren kalitatea (FMeasure) handiagoa delako eta denbora gutxiago eramaten duelako.
- Esparru esperimentaleko atalean ateratako emaitzak aztertuta erabaki da hurrengo konfigurazioa erabiltzea:
  - Sailkatzailea: RandomForest.
  - Errepresentazio espazioa: Bag of Words .
  - Atributu hautapena: Bai.

  Hau guztiarekin 0.8803eko FMeasurea lortu dugu 8.4698 segundutan.

Proiektuaren helburuak lortu dira, hau da, bezeroak (kasu honetan irakasleak) eskatutako guztia egin da eta hauek dira proiektu honen ahuleziak eta sendotasunak:

- Sendotasunak:
  - RandomForest eta OneRule algoritmo sailkatzaileak nahiko sinplea direnez, oso azkar prozesa ditzakegu instantzia eta atributu kopuru handiak.
  - Kalitate/Denbora baremoa oso ona da, kalitate "handia"ematen du denbora gutxitan.

- Ahuleziak:
  - RandomForest ez da sailkatzaile oso aproposa testu meatzaritza egiteko.
  - OneRule ez da sailkatzaile oso aproposa testu meatzaritza egiteko.
  - Instantzia gehienak spam ez direnez, iragarpen gehienak ondo egiten ditu (%97 baino iragarpen gehiago ondo egiten ditu), baina spam-en FMeasure-a ez da hain ona (0.8803).

Honetaz aparte proiektua egiteko edukitako denbora oso mugatua izan da, denbora gehiago izatekotan hurrengo elementuak inprementatuko litzateke:

- Prozesu guztiak hodeian egitea eraginkortasuna handitzeko.
- Programa web orrialde batean jartzea handik exekutatzeko bezeroari erabilera asko errazteko eta edozein ordenagailutik, mugikorretik edo tabletik erabili ahal izateko.
- Beste sailkatze algoritmo ezberdinekin frogatu, adibidez sare neuronal batekin.

Bukatzeko proiektua berriro zeroz hasiko bagenu lehenengo inplementazio honetan ateratako esperientzia bideratuko genuke froga gehiago egitera eta aurreko parragrafoan adierazten denaren arabera azken bezeroari erraztasunak ematera programa era egokian erabiltzeko. Honetaz aparte lana garatzerakoan ikusi egin dugu RandomForest ez dela oso sailkatzaile ona testurako, honen ondorioz, beste sailkatzaile hobeago bat bilatuko dugu.

6 Bibliografia

[1] Niladri Biswas. Text mining and its business applications.

[2] Jason Brownlee. A gentle introduction to the bag-of-words model.

[3] Jason Brownlee. How to use ensemble machine learning algorithms in weka.

[4] Jiakang Chang, Christian O' Reilly, Nancy Pontika, Gareth Owen, Kenneth Haug, and Martine Oudenhoven. ¿qué es la minería de textos, cómo funciona y por qué es útil?

[5] Aurelio Germes. ¿qué es un vector tf-idf?

[6] Bartosz Góralewicz. The tf*idf algorithm explained.

[7] University of Waitako. Class attributeselection.

[8] University of Waitako. Class oner.

[9] University of Waitako. Class paralleliteratedsingleclassifierenhancer.

[10] Dr. Saed Sayad. Oner.

[11] Thinrhino. Calculate tf-idf.

[12] Wikipedia. Bootstrap aggregating.

[13] Tony Yiu. Understanding random forest.