

ERABAKIAK HARTZEKO EUSKARRI SISTEMAK

Hirugarren maila, 31. taldea

2. Lauhilabetea

Suicide Ideation - Random Forest

Aingeru Bellido, Nagore Gómez, Sergio Martín eta Maitane Urruela.

Bilbon, 2023ko martxoaren 31a

Aurkibidea

1	Proiektuko atazak eta arduren banaketa	2
2	Oinarri teorikoa	3
2.1	<i>Baseline</i>	3
2.2	Atributuen espazioa: <i>Bag of Words</i>	3
2.3	Atributuen hautapena informazio irabazian oinarrituta	4
2.4	Random Forest eta parametro ekorketa	5
2.5	Eredua eta estimatutako kalitatea	8
2.6	Sailkapena	8
3	Oinarri esperimentalak	9
3.1	Datuak	9
3.2	Aurre-prozesamendua	9
3.2.1	Datuen formatua: <i>ARFF</i>	9
3.2.2	Atributuen espazioa: <i>Bag of Words</i>	11
3.2.3	Atributuen hautapena informazio irabazian oinarrituta	12
3.3	Ereduren inferentzia eta itxarondako kalitatearen estimazioa	14
3.3.1	Eredu sailkatzailearen parametro ekorketa	14
3.3.2	Eredua eta estimatutako kalitatea	18
3.4	Sailkapena	19
3.5	Emaitza esperimentalak	20
3.6	Emaitzen diskusioa	20
3.7	Exekuzio adibidea	22
3.7.1	Script-en bidezko exekuzioa	22
3.7.2	Exekutagarrien bidezko exekuzioa	23
4	Ondorioak eta etorkizunerako lana	26
5	Balorazio subjektiboa	27
6	Bibliografia	28

1 Proiektuko atazak eta arduren banaketa

Gure taldeari esleitutako ataza *Suicide Ideation* izan da. Ideazio suizida, edo pentsamendu suizidak, norberaren bizitza amaitzeko aukerari buruzko ideiak edo zurrumurruak izateko pentsamendu-prozesua da. Ez da diagnostikoa, buru-nahaste batzuen sintoma baizik, eta gertaera kaltegarriei erantzuteko modu bezala ere agertu daiteke, buru-nahasmendurik izan gabe.

Suicide Ideation-ak Interneten pertsonak jartzen dituzten mezuen analisisian datza, hauek aztertzen dira eta pertsona horrek bere buruaz beste egiteko ideia duen iragartzeko erabiltzen da.

Proiektua lantzeko *RandomForest* algoritmoa erabiliko da, lortutako datuetatik (mezuak, tweet-ak...) buruaz beste egiteko kasuak iragartzea lortu nahi da.

Proiektua Java lengoaia erabiliz burutuko da, IntelliJ tresna eta Weka paketearekin.

Atalak tadekideon artean banatuko dira, taldekide bat atal baten arduraduna izan arren, taldekide guztien artean garatu izan da atal hori. Ataza nagusien arduren banaketa hurrengoa izanik:

Aingeru: *RandomForest* eta parametro ekorketa

Nagore: Atributuen espazioa (*BoW*)

Sergio: Datuen formatua eta aurreprozesamendua

Maitane: Atributuen hautapena (*FSS*)

2 Oinarri teorikoa

2.1 *Baseline*

Garatutako atazaren kalitatearen behe estimazioari *baseline* deritzo. *Baseline* delakoak ataza garatzerakoan espero den behe bornea adierazten du. Gehienetan, behe borne hau lortzeko eredu sinplea eta konputazio behar txikikoa erabiltzen da, gero modelo optimoa lortzeko erabiliko den klasifikadorearen familia berekoa.

Kasu honetan zuhaitzak egiten dituen *RandomForest* klasifikadorea erabiliko denez, hurrengoak egongo lirateke aukeren artean: *RandomForest* bera, *J48*, etab. Kasu honetan *RandomForest* erabili da defektuz dakarren parametroekin, izan ere, proba azkarra izan behar du eta ezin da itxaron denbora luzez emaitzarik pesimistena lortzeko. Ez litzateke zentzukoa izango eredu konplexu bat erabiltzea *baseline* lortzeko.

Neurtu nahi dena *baseline* eta eredu konplexuak ematen dituen emaitzen arteko aldakuntza da. Hau da, zenbateko hobekuntza suposatzen duen eredu iragarlearen inferentzia prozesuak. Hau txikia edo negatiboa izanda, ez litzaketelako zentzuzkoa prozesu osoa burutzea horrek suposatuko lukeen kostearekin.

2.2 Atributuen espazioa: *Bag of Words*

Datuen instantzia bakoitzak testu bat, kasu honetan sare sozialetako mezua, errepresentatzen du eta atributu bakar baten bidez, testua, eta klasearen bidez karakterizatuta dator. Puntu honetan testuaren atributua zenbakizko bektore bihurtu nahi da.

Zenbakizko bektorearen dimentsioa instantzietan dauden hitz kopuruarekin bat etorriko da, baita sortuko den hiztegiaren tamainarekin ere. Modu honetan, instantziek atributuen agerpena adieraziko dute.

Instantzien agerpena irudikatzeko hainbat modu daude. Alde batetik, *BoW*, hitzen presentzia/ausentzia adierazten duena, edo *TF-IDF* (*BoW*-ren aldaera), hitz bakoitzaren adierazgarritasuna dokumentu multzoan kuantitatiboki determinatzen duena. Bestetik, *Sparse* (dispertso), zero asko dituen matrizea, edo *NonSparse* zerorik ez duen matrizea. Hau da, matrize dispertsoko lerro bakoitzean hiztegiak dituen hitz adina elementu agertuko dira, instantzia horretan ageri direnak 1 batez adierazita eta ageri ez direnak 0 batez adierazita. Aldiz, matrize ez dispertsokoan soilik agerpena eta atributu bakoitzaren maiztasuna aurkezten da. Haatik, kontuan hartu beharrekoa da Wekak alderantzizko deitura erabiltzen duela.

2.3 Atributuen hautapena informazio irabazian oinarrituta

Bag of Words lortzen denean hasierako testuetatik, atributu kopurua testu guztietan agertutako hitz guztien bilduma osatzen du. Kontuan izanda milaka instantzia daudela eta horietako testu bakoitzeko dozenaka hitz egon daitezkeela (noski testuen artean errepikaturik dauden hitzak ere agertuko dira), atributuen multzoa nahiko handia da. Multzo honetako hitzak (hauen presentzia ala ausentzia) atributu bezala hartuz, erabaki espazioaren dimentsioa oso altua izatea dakar.

Instantziak errepresentatzeko atributu kopuru altuarekin alderatuz, hauen agerpena instantzia bakoitzeko ratioa baxua geratzen da. Hori dela eta, eta honek izan ahal dituen kalteak saihesteko dauden atributu guzti hauetatik azpi-multzo bat aukeratzea komeni da, hau da, atributuen hautaketa burutzea.

Hau egiteko atributuen hautapen teknikak erabiltzen dira, atributu erredundanteak edo garrantzia gutxikoak baztertuz, eta beraz, bilaketa espazioaren dimentsioa murriztuz. Hala ere, hautapen hau egitea informazio galera ekar dezake, beraz, murriztapena kontrolatua izan behar du.

Proiektu honetan informazio irabazian oinarrituta egin da hautaketa hori, hain zuzen ere, ebaluatzaile bezala *InfoGainAttributeEval* erabiliz.

Honek, datu-multzo bateko atributu bakoitzaren garrantzia zehazteko, informazioaren teorian eta entropian oinarritzen da. Honen bidez neurtzen du atributu bakoitzak klase aldagaiari dagokionez ematen duen informazio kopurua.

Behin informazio kopuru hori neurtuta, ebaluatzaileak garrantziaren arabera ordenatutako atributuen zerrenda bat sortuko du. Zerrendaren hasieran klaseari ekarpen gehien suposatzen dioten atributuak egongo dira, eta behera joan ahala, ekarpen txikiagoa suposatzen diotenak.

Garrantzi txikiko atributuak kenduz, ereduaren konplexutasuna murriztu eta datu berriak orokortzeko ahalmena hobetuko da modeloan.

2.4 Random Forest eta parametro ekorketa

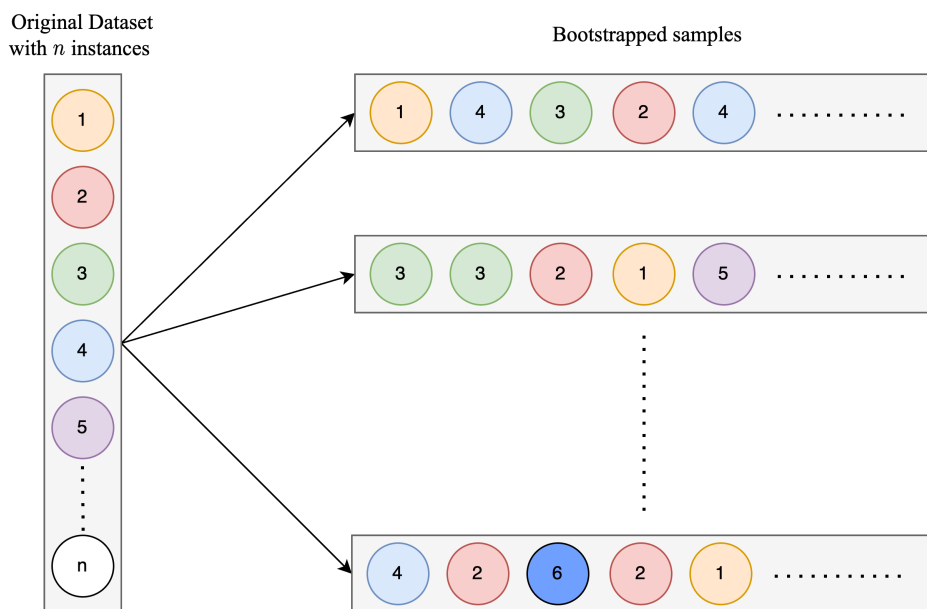
Atal honetan, *RandomForest* sailkatzailearen inferentziaren buruzko azalpena egingo da. Gainera, honek erabiltzen dituen parametroak deskribatuko dira eta zeri egiten dioten erreferentzia.

RandomForest algoritmoa (Breiman, 2001) erabaki-zuhaitzak erabiltzen dituen sailkatzailea da. Ikaskuntza gainbegiraturako teknika bat da, eta hainbat erabaki-zuhaitz sortzen ditu entrenamenduko datu-multzo baten gainean. Lortutako erabaki-zuhaitz guztiak konbinatu egiten dira, eredu bakar sendoagoa lortzeko.

Algoritmoak hurrengo pausuak egiten ditu ikasketa prozesuan:

1. **Bootstrapping prozesua** (Irudia 1): Emandako datu-sorta erabiliz, ausazko laginak lortzen dira. Lagin hauek, ordezkapenarekin egiten dira, hau da, lagin bakoitzean, instantzia bat behin baino gehiagotan ager daiteke. Prozesu honi *bootstrapping* deritzo eta lagin bakoitzari *bootstrapped sample*. Teknika honi esker, lagin bakoitza bestetikiko desberdina izatea lortzen da.

Hartzen diren lagin kopurua **numIterations** parametroak zehazten du Weka-n eta lagin bakoitzean hartzen diren instantzia kopuruaren portzentaia, instantzia totalen gainean, **bagSizePercent** parametroak zehazten du.



Irudia 1: *Bootstrapped sampling*

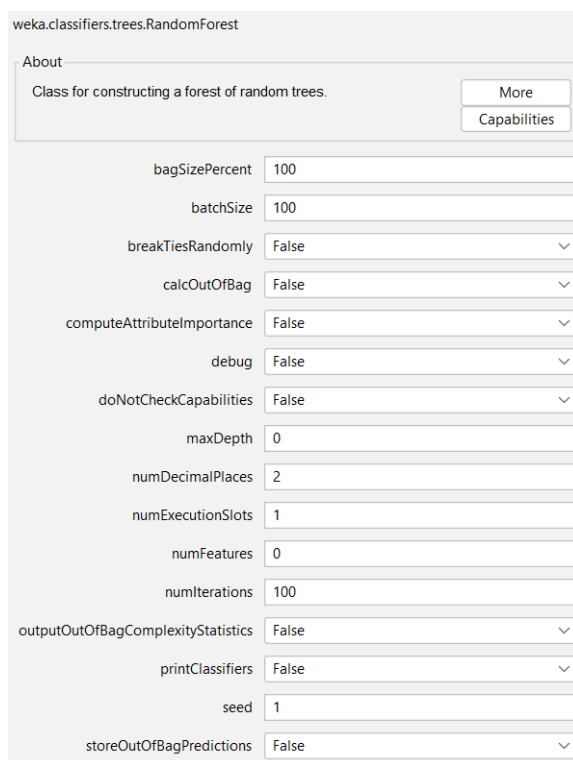
2. **Erabaki zuhaitzak sortu**: Algoritmoak erabaki-zuhaitz bat sortuko du hautatutako lagin bakoitzarentzat. Horretarako, hainbat parametro erabiltzen ditu zuhaitzen ezaugarriak lortzeko. Zuhaitz bakoitzaren sakonera maximoa zehazteko, Weka-k **maxDepth** parametroa erabiltzen du.

Bestalde, *numFeatures* parametroaren bitartez, zuhaitz bakoitza sortzeko erabiliko diren atributu kopurua zehaztuko da. Balio hori kontuan hartuta, zuhaitz bakoitzerako atributuak ausaz aukeratuko dira, hau da, zuhaitz bakoitzak atributu desberdinak erabiliko ditu bere zuhaitz propioa sortzeko baina guztiek erabiliko dute atributu kopuru bera.

Beraz, *Random* ezaugarria bi kasutan aplikatzen da, *bootstrapping* egitean eta lagin bakoitzerako ausazko atributu hautapena egiten denean. *Random* prozesu hau hainbeste kasutan egiten denez, entrenamendu fase hau oso egonkorra izatea lortzen da, hainbat laginekin eta kasu desberdinekin entrenatzen delako sailkatzailea.

Bestalde, *Forest* kontzeptua, erabaki-zuhaitz multzotik dator.

Weka-k hainbat parametro ezberdin konfiguratzeko aukera ematen du (Irudia 2).



Irudia 2: Weka-ren *RandomForest* sailakatzaillearen parametroak eta balio lehenetsiak

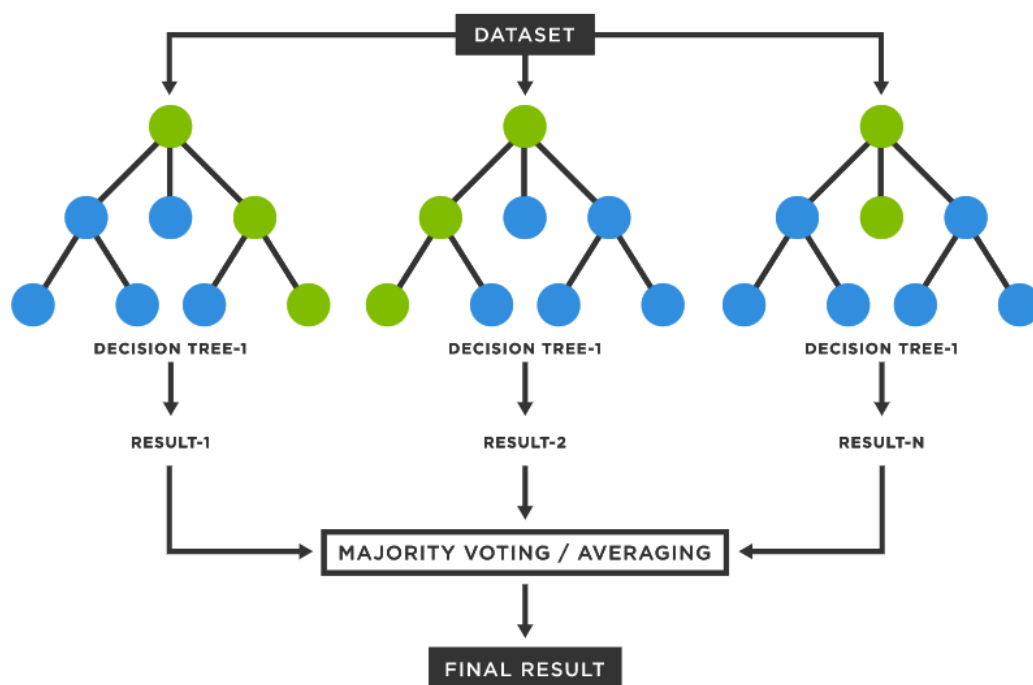
Parametroak hainbeste badira ere, sailkatzailearekiko esanguratsuenak eta optimizazio prozesuan emaitzarekiko sentikorrenak direnak erabili dira, lehen aipatutakoak, *numIterations*, *bagSizePercent*, *maxDepth* eta *numFeatures*, hain zuzen ere.

Algoritmoak hurrengo pausuak egiten ditu iragarpenak egiteko:

1. **Iragarpenak:** Klasea iragarri nahi den instantzia bakoitzeko, instantzia hori sortutako erabaki-zuhaitz bakoitzarekin iragarri egiten da eta zuhaitz bakoitzak ematen duen emaitza gordetzen da.

2. **Aggregation:** Aurreko pausuan gordetako emaitza bakoitzeko, emaitza bakar bat sortzen da instantziaren klasea iragartzeko. Klasifikazio problema bat bada, iragarpenera jasotako emaitza guztien moda izango da; bestela, erregresio problema bat bada, jasotako emaitzen batezbestekoa.

Beraz, azaldutako prozesu osoa, hurrengo eskemaren (Irudia 3) bitartez adieraz daiteke:



Irudia 3: RandomForest laburpen prozesua

Azaldutako prozesu osoari, *Bagging* (*bootstrap aggregating*) deritzen. Honi esker, eredu sailkatzailaren egonkortasuna mantentzea lortzen da.

2.5 Eredua eta estimatutako kalitatea

Ereduaren kalitatea estimatzeko hurrengo hiru eskemak erabili dira:

- Ebaluazio ez-zintzoa (*resubstitution error*)

Eredua ebaluatzen da entrenamendurako erabili den multzo berarekin. Modu honetan, estimatutako kalitatearen goi borneya lortuko da, baina ez da kalitatearen estimazio errealista izango, baizik eta optimistena.

- *10-fold Cross Validation*

Ebaluaketa egiteko multzo berdina K *folds*-etan banatzen da, kasu honetan, 10 *folds*-etan. Hau ez da zintzoa izango, izan ere, entrenamendurako erabili izan diren datu berdinak erabili izan dira. Dena den, ebaluazio ez-zintzoa baino emaitza errealistagoak emango ditu (eta beraz, ez hain optimistak).

- *Repeated Hold-out* estratifikatua

Hold-out-en aldaera da estratifikatua, gainbegiraturako multzoa bitan banatzen da (*train* eta *dev*), kasu honetan, klase-banaketa mantenduz. Instantzia guztien %70 erabili da *train* multzorako eta gainerakoa *test* multzorako. Prozesu hau hainbat aldiz errepikatzen da (kasu honetan 5) emaitzen batezbestekoa ateratzeko (hurbilpen errealistagoa). Honek adieraziko du modeloak duen borneya errealista eta gero iragarpenetan agertzea espero dena.

Prozesu hau *baseline* zein modelo optimoarekin burutzen da. Hemen lortutako emaitzak konparatuko dira prozesu osoa burutzeak zentzuzkoa den erabakitzeke. Lehen esan den moduan, espero da modelo optimoaren bidez lortutako emaitzak positiboagoak izatea.

2.6 Sailkapena

Bi modelo lortuta, *Baseline.model* eta *RandomForest.model*, hain zuzen ere, test berdinarekin bi iragarpen burutuko dira modelo bakoitzarekin.

Beraz, *baseline*-tik lortutako emaitzak txarragoak izatea espero da, biak *RandomForest* sailkatzaileak izanda, honek defektuzko parametroak erabiltzen baititu, besteak prozesu osoaren bidez lortu izan diren parametro optimoagoak erabiltzen dituen bitartean.

Proiektu osoaren helburua sailkapena da eta behin modeloa lortuta (prozesuaren emaitza), honek izango duen lana, baina dauden instantzia gainbegiratuak sailkatu beharrean, klasifikatu gabekoak sailkatuko dira.

3 Oinarri esperimentalak

3.1 Datuak

Proiektua egiteko hainbat datu multzo ezberdinen artean *"Suicide and Depression Detection"* datu sorta aukeratu da, izan ere, pertsonen joera suizidak ikusteko adierazgarriena da. Datu hauek, 232074 instantzia daude, bakoitza hiru atributu dituenak: instantziaren ID-a, *tweet*-ean agertzen den testua eta klasea.

ID-a, atributu numerikoa da, informazio gehigarriak ematen ez duena; bakarrik *tweet*-a identifikatzeko balio du. Bestetik, testua *string* motatako atributua da eta dituen balio guztiak bakarrik dira, hau da, ez da *tweet*-ik errepikatzen. Azkenik, klasea bi balio nominal izan ditzake: *'suicide'* (testua idatzi duen pertsona bere buruaz beste egin bazuen) eta *'non-suicide'* (kontrakoa).

Datuetatik ere esan daiteke hasiera batean *tweet*-ak klasearekiko orekatuta daudela, esate baterako, instantzia kopuru bera daude *'suicide'* eta *'non-suicide'* klasekoak (bakoitzak 116.037 instantziekin). Beraz, ez dago klase minoritariorik hasierako datu multzo honetan.

Azkenik, instantzietan ez dira *missing value* balioak dituzten instantziarik agertzen.

3.2 Aurre-prozesamendua

3.2.1 Datuen formatua: ARFF

Eredu sailkatzailearen inferentziarako erabiliko diren datuak webgunetik deskargatu dira *CSV* formatuan. Wekak *ARFF* formatua baino ez du onartzen datuen prozesamendua egiteko, beraz, hasierako datuak Wekak ulertu ditzan formatu egokitzapena egin behar da. Horretarako, *getArff* metodoa garatu da, *CSV* fitxategi bat jasota *ARFF* formatuko fitxategi bat bueltatzen duena.

Wekak berez metodo pare bat du *CSV*-tik *ARFF*-rako eraldaketa burutzeko era laburrean, hurrengo kodea erabiliz hain zuzen ere:

```
//Load CSV
CSVLoader loader = new CSVLoader();
loader.setSource(new File(args[0]));
Instances data = loader.getDataSet();

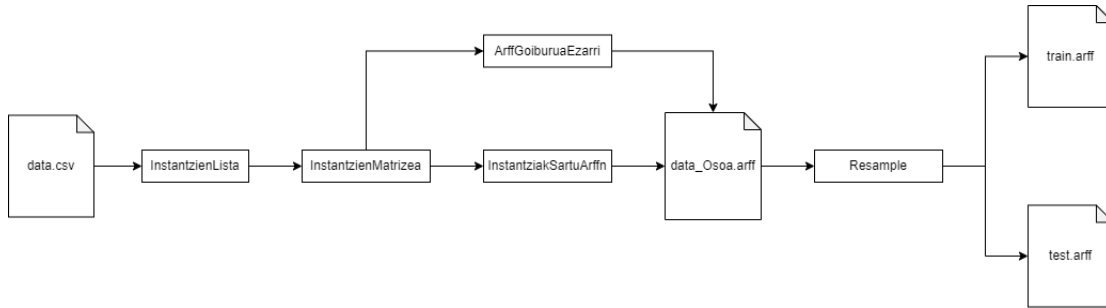
//Save Arff
ArffSaver saver = new ArffSaver();
saver.setInstances(data);
saver.setFile(new File(args[1]));
saver.setDestination(new File(args[1]));
saver.writeBatch();
```

Dena den, proiektu honetan hasieran jasotako datuen *CSV* formatua konplexua zen eta berrehun mila instantzia baino gehiago izanda (milioi bat lerroko *CSV*-a gutxi gorabehera), aldatzeko zaila

ere. Hori dela eta, eskuz egindako transformazio bat planteatu izan da kasu honetarako.

ARFF-ak lortzeaz gain, *getArff* (Irudia 4) metodoan datuen garbiketa egiten da. Garbiketaren helburu nagusia instantzia guztiak Wekarentzat interpretagarriak izatea eta esanguratsuenak ez direnak kentzea da. Ondorengo iragazkiak erabili dira instantziak filtratzeko:

- 70 karaktere baino gutxiago eta 1500 karaktere baino gehiago dituzten instantziak deuseztatu
- 20 karaktere baino gehiago dituzten hitzak kendu
- Karaktere bakarreko hitzak kendu
- Soilik karaktere bereziak dituzten instantziak ezabatu
- Instantzietan dauden puntuazio markak eta zenbakiak kendu
- *Emojiak* testu bihurtu (testuak emojiaren deskribapena islatzen du)
- Braille hizkuntzako ikurrak ezabatu
- Instantzietan komillak eta *enter*-ak kendu



Irudia 4: *GetArff* metodoaren diagrama.

Behin garbiketa hau egin denean, *trainRAW.arff*, *testRAW.arff* eta *devRAW.arff* fitxategiak lortzen dira (Taula 1).

	Raw data		
	Train	Dev	Test
Nominal Atributes	1		
Numeric Atributes	0		
String Atributes	1		
Boolean Atributes	0		
Atributes total	2		
Instances +	26081	7830	22276
Instances -	29219	8760	25111
Instances total	55300	16590	47387

Taula 1: Jatorrizko datuen deskribapen kuantitatiboa.

3.2.2 Atributuen espazioa: *Bag of Words*

Behin datuen garbiketa eginda, *train* eta *dev* lortzeko *Resample* filtroa erabili da, klase-banaketa mantentzeko edo banaketa uniformerantz bideratzeko. *Train*-erako instantzien %70 eta gainerakoa *test*-erako.

Train multzoaren bektore errepresentazioa lortzeko ***StringTo WordVector*** filtroa erabili dugu, honek *String* atributuak testuetako hitzen agerpenari buruzko informazioa adierazten duten zenbaki-atributuen multzo bihurtzen ditu.

Hiztegi bat sortzen du *setDictionaryFileToSaveTo* *train*-aren lehenengo datu-multzotik, baina aipatu beharra dago filtroa ez dela erabat gainbegiratuta klase-atributua ezartzen denean, klase bakoitzerako hiztegi bereizi bat sortzen duelako eta gero konbinatzen dituelako.

Mantenduko diren atributu kopurua ezartzeko, *setWordsToKeep* ezarriko da, gure kasuan 10000 atributu ezarri dira (gutxi gorabehera 18000 lortu dira).

Bektorearen errepresentazio ezberdinak lortu daitezke:

- *BoW* erabiltzen bada, *setOutputWordCounts true* ezarri behar da, *i* instantziako *j* posizioan dagoen atributuaren edukia bitarra izateko, *j*. hitza *i*. mezuan ausente (0z adierazita) ala presente (1ez adierazita) dagoela adierazten du.
- *TF-IDF* erabiltzen bada, *setOutputWordCounts false*, *i* instantziako *j* posizioan dagoen atributuaren edukia, *j*. hitza *i*. mezuan duen agerpen maiztasuna adierazteko.

Horietaz gain, *Sparse* edo *NonSparse* nahi den aukeratu beharko da, *NonSparse* nahi izatekotan, ***SparseToNonSparse*** filtroa aplikatu beharko da, modu honetan matrize dispertsoa lortuko da (Weka-rentzat ez-dispertsua).

BoW edo *TF-IDF* eta *Sparse* edo *NonSparse* aukerak frogatu dira gutxi gora behera 20000 instantziekin ebaluaketa eta iragarpenak eginez (Taula 2). Nahiz eta ebaluaketa eta iragarpenak kasu guztietan datu multzo berdinarekin egin diren, emaitzek ezberdintasun txikiak dituzte.

	Correctly Clasified Instances (%)			
	Ez zintzoa	10-fCV	Stratified Repeated Hold Out	Iragarpen asmatze-tasa
BOW+Sparse	100	87.82	87.98	87.20
BOW+NonSparse	99.98	87.36	86.69	87.38
TF-IDF+Sparse	99.98	87.31	86.45	87.41
TF-IDF+NonSparse	99.98	87.43	87.46	87.43

Taula 2: *BoW* edo *TF-IDF* eta *Sparse* edo *NonSparse* erabiliz ebaluaketa eta iragarpenen emaitzak.

Atributuen hautapena egin ondoren, behin aukeratutako atributuak soilik dituen *trainFSS.arff* eta hiztegi berria *hiztegiaFSS.arff* lortuta, *test* eta *dev* multzoak bateragarriak egin behar dira, geroago ebaluaketa eta iragarpenak egin ahal izateko.

Hori lortzeko **FixedDictionaryStringToWordVector** filtroa aplikatu zaio multzoari, **StringToWordVector** filtroaren gauza bera egiten du, baina hiztegia ezarri behar zaio, kasu honetan aipatutako *hiztegiaFSS.arff*, *trainFSS.arff* fitxategiaren bateragarria den *testFSS.arff* eta *devFSS.arff* lortzeko.

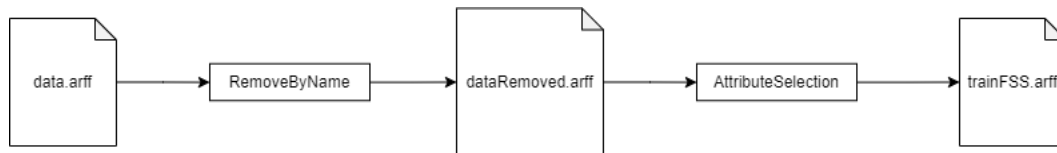
Datuen *BoW* errepresentazioa lortzean lortutako datuak taula batean antolatu dira (Taula 3).

	BOW		
	Train	Dev	Test
Nominal Atributes	1		
Numeric Atributes	18777		
String Atributes	0		
Boolean Atributes	0		
Atributes total	18778		
Instances +	26081	7830	22276
Instances -	29219	8760	25111
Instances total	55300	16590	47387

Taula 3: *BoW* datuen deskribapen kuantitatiboa.

3.2.3 Atributuen hautapena informazio irabazian oinarrituta

Orain, hasierako instantzietako testuetan zeuden hitz guztiak atributu bezala lortu dira *train-BoW.arff* fitxategian. Lehen azaldu den moduan, honek erabaki espazioaren dimentsioa oso handia izatea suposatzen du, beraz, atributu selekzioaren bidez dimentsio hori murriztuko da. Diagrama baten bidez adierazi dira jarraitu izan diren pausuak (Irudia 5).



Irudia 5: *FSSInfoGain* metodoaren diagrama.

Lehenik eta behin, **RemoveByName** filtroa erabiliz, hitz arrunten egoera adierazten ez duten atributu guztiak kendu dira (espazioak, ikono arraroak, etab.), izan ere, ez dute baliozkoa den informaziorik ematen, haien agerpena oso murriztua eta esanahi gutxikoa baita. Hau egin ostean, geratu diren atributuen artean benetako hautapena burutu da.

Proiektu honetan atributuen hautapena informazio irabazian oinarritua dago, Weka-ko **AttributeSelection** filtroaren ebaluatzaile bezala *InfoGainAttributeEval* erabiliz. Bestetik, atributuen azpimultzoaren bilaketa metodoa bezala *Ranker* erabili da.

Lehen azaldu den moduan, *InfoGainAttributeEval* ebaluatzaileak dauden atributu guztien zerrenda bat sortzen du, hauek klaseari suposatutako informazio irabaziaren arabera ordenatuz. Honen bidez ez da atributurik kenduko, baina garrantziaren arabera orden bat lortuko da.

Lista honetatik atributuak ezabatuko dituen *Ranker* da. Azkenengo honek 2 parametro ditu: *numToSelect*, emandako atributuetatik mantendu nahi diren atributu kopurua adierazten duena eta *threshold*, atributu bat ez baztertzeko behar duen balio minimoa (defektuz *long integer* txikiena).

Bi parametro hauekin hainbat konbinazio frogatu ostean, atera diren ondorioen artean esanguratsuenak da *threshold* balioa ez duela eraginik gure datuen gainean. Hau ondorioztatu izan da, *long integer* txikienetik handienara joanez hainbat balio hartuta, beti lortu izan direlako emaitza berak.

Bestetik, *numToSelect* parametroarekin ikusi da nola gero eta parametro gehiago hartuz, lortzen den *weighted F-Measure* balioa tendentzia negatiboa hartzen duen. Hori dela eta 2000 balioa eman zaio, handiegia ez dena ditugun atributu kopurua kontuan izanda.

Atributuen hautapena amaitzerakoan lortutako datuak taula batean adierazi dira (Taula 4).

	BOW and FSS		
	Train	Dev	Test
Nominal Atributes	1		
Numeric Atributes	2000		
String Atributes	0		
Boolean Atributes	0		
Atributes total	2001		
Instances +	26081	7830	22276
Instances -	29219	8760	25111
Instances total	55300	16590	47387

Taula 4: Aurre-prozesatutako (*BoW* eta *FSS*) datuen deskribapen kuantitatiboa.

3.3 Ereduaren inferentzia eta itxarondako kalitatearen estimazioa

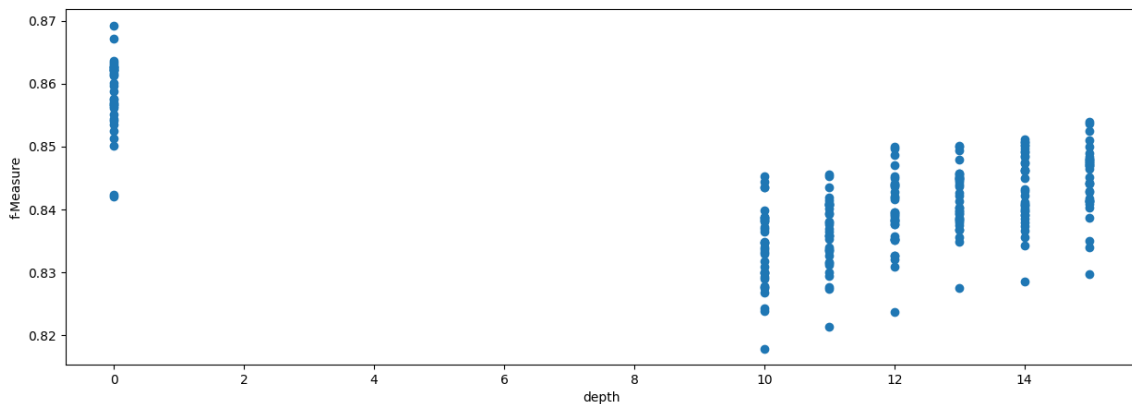
3.3.1 Eredu sailkatzailearen parametro ekorketa

2.4 atalean aipatu den bezala, *RandomForest* eredu sailkatzailearen *numIterations*, *bagSizePercent*, *maxDepth* eta *numFeatures* parametroen ekorketa egin da. Ekorketa egiteko, klase minoritarioaren *F-measure* ebaluazio metrika optimizatzea izan da helburua. Horretarako, estratifikatutako *train* eta *dev* jasota, *hold-out* bat gauzatu da.

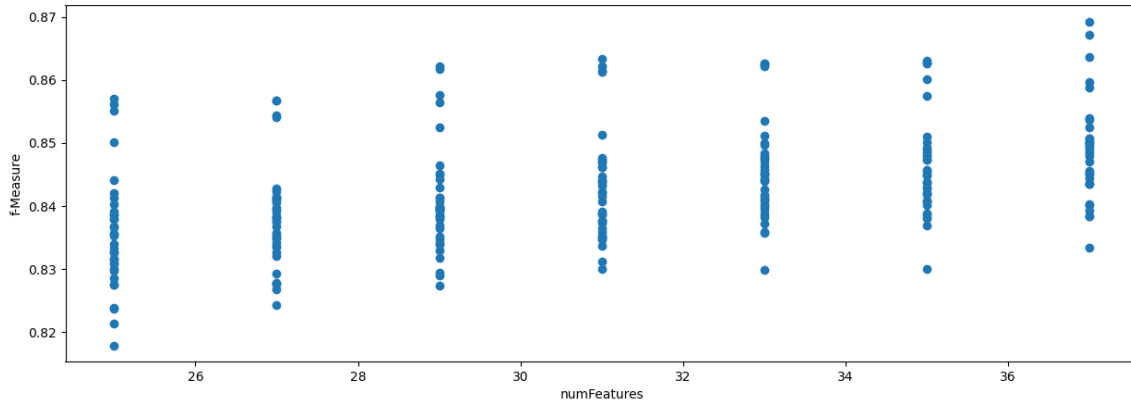
Denbora eta konputazio ahalmen mugatuak direla eta, instantzien multzoa eta ekortutako parametro kopurua murriztu behar izan da, epearen barruan emaitzak lortzeko. Horregatik, parametro ekorketaren prozesua eskuragarri dauden instantzien %5-arekin egin da, 10000 instantzia inguru. Izan ere, konputazio ahalmen handia behar da instantzia eta atributu askoren *RandomForest* sailkatzaile bat entrenatzeko.

Arestian aipatutako denbora eta konputazio baliabide murrizketak egon izan ez balira, instantzia multzo handiagoarekin lan egitea egokiagoa izango litzateke. Gainera, parametro multzo bakoitze-ko *repeated hold-out* ebaluazio eskema erabiltzeak ere emaitzak optimizatzea ekarriko luke. Hala ere, *RandomForest*-en *Bootstrapping* prozesuan hainbat partiketa sortzen dira. Beraz, lortutako emaitzak nahiko errealistak izan dira nahiz eta *hold-out* sinplea erabili.

Gauzak horrela, *maxDepth* eta *numFeatures* parametroetan jarri da esfortzu gehiena. Izan ere, hauen optimizazioa sailkatzailearen kalitatea gehien hobetzen duten parametroak dira zenbait proba jorratu ostean, hurrengo irudietan (Irudia 6 eta Irudia 7) ikusten den bezala.



Irudia 6: *maxDepth* parametroaren grafikoa

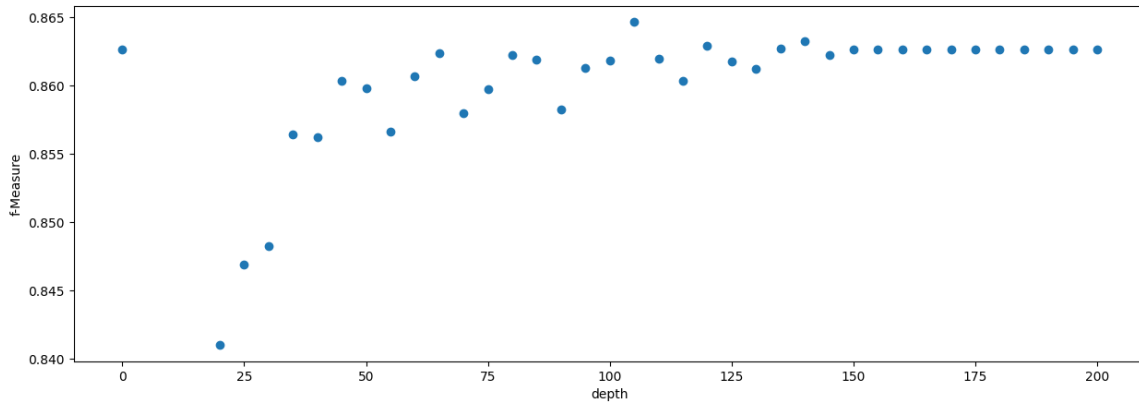


Irudia 7: *numFeatures* parametroaren grafikoa

(Oharra: azken bi grafikoen (Irudia 6 eta Irudia 7) ekorketa instantzien %5 baino handiagoarekin egin da eta *F-Measure* balioak ezin dira zuzenean konparatu aurrerantzean aipatuko diren balioekin.)

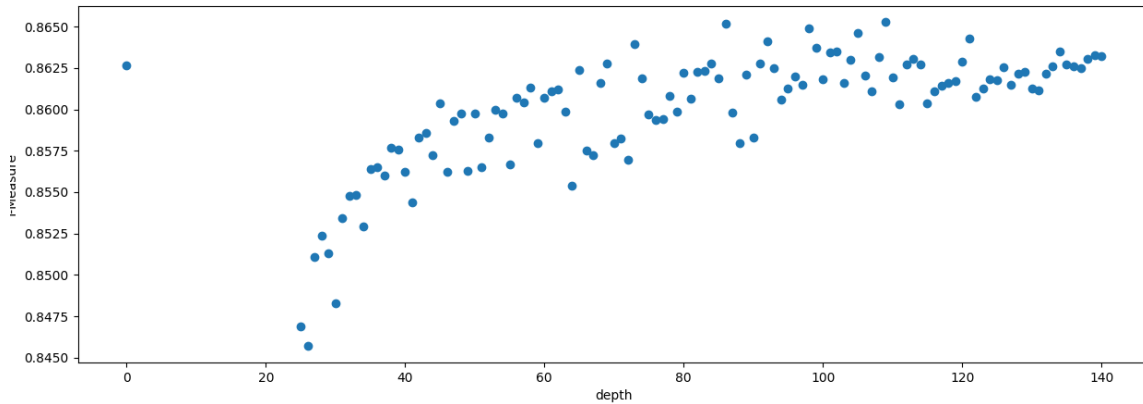
Beraz, ikusita parametro hauen balioa zenbat eta gehiago handituz lortzen diren balioak asko hobetzen direla, hauetan esfortzu gehiena jarri da. Hasiera batean, parametroak banan-banan ekortu dira beste parametroak optimoaren oso hurbil dauden balioak erabiliz, hots, Weka-k defektuz erabiltzen dituen parametroen balioen oso antzekoak.

maxDepth parametroa ekortzeko, kontuan izan behar da Weka-k 0 balioa *unlimited* bezala hartzen duela, hau da, 0 balioa balio oso-oso altuen parekoa izango da. Horregatik, lehen proba bat egin da (Irudia 8) 0 balioarekin eta balio tarte oso handi batekin.



Irudia 8: *maxDepth* parametroaren tarte handia ekortzen

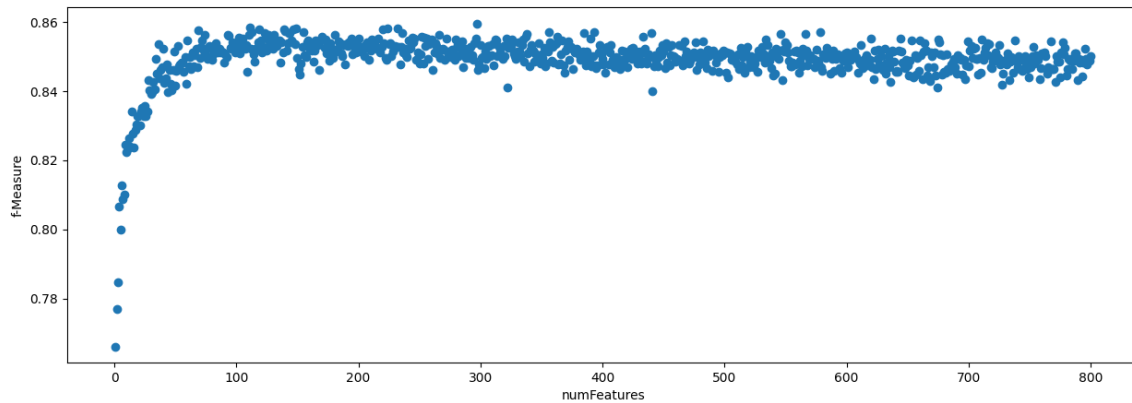
Adierazgarri den bezala, >150 diren balioetan, emaitzak nahiko antzekoak dira eta espero genuen moduan, 0 balioarekin lortzen diren antzeko emaitzak. Argi ikusten da balio optimoa 75 eta 140 artean dagoela. Horregatik balio tarte txikiago batean ekorketa (Irudia 9) zehatzago bat gauzatu da.



Irudia 9: *maxDepth* parametroaren tarte txikia ekortzen

Azken ekorketa honetatik, balio optimoa 109 dela lortu da. Espero zen bezala, erabaki-zuhaitzek zenbat eta sakonera handiagoa izan, orduan eta erabaki zehatzagoak hartzen dute baina balio handi batzuetatik aurrera balio konstanteak lortzen dira. Balio handi horiek, optimoaren hurbileko balioak izaten jarraitzen dute baina ez dira hain baliagarriak, sailkatzailearen entrenamendua konputazio-nalki izugarri handitzen delako, batez ere.

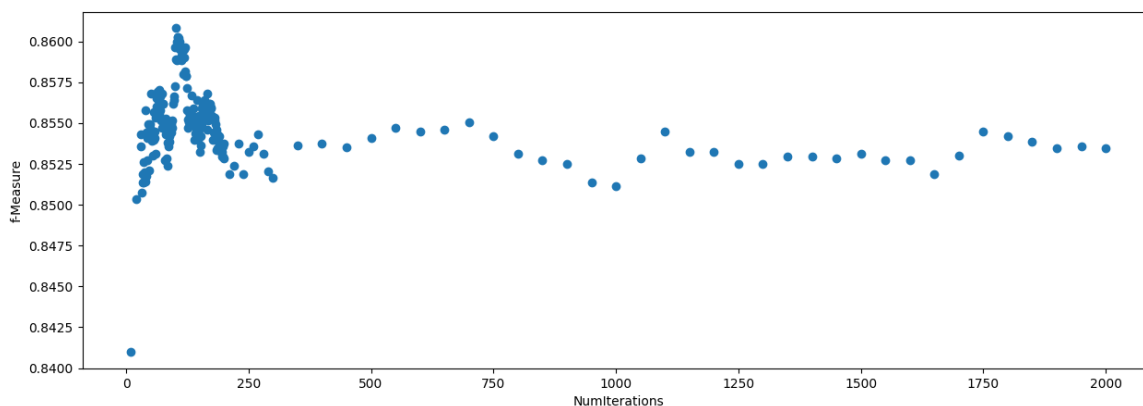
numFeatures parametroa ekortzerakoan (Irudia 10) hurrengo emaitzak lortu dira.



Irudia 10: *numFeatures* parametroa ekortzen

Ikusten den bezala, *numFeatures* parametroaren balio txikiak ekortuz berehala hobetzen dira emaitzak. 110 balioetatik aurrera, emaitzak txikiagotuz doaz. Ekorketa honek, 297 balio itzuli du parametro optimo bezala. Balio hau, tendentzia beherakorra duen balio tarte batean lortu da. Kasu honetan, kontuan izan behar da beharbada 100 inguruko balioak erabiltzea beharrezkoa izan daitekeela eredu sailkatzailearen inferentzia konputazio ahalmen txikiagoa izan dezan. Bestela, entrenamendu fasean, atributu asko aukeratu behar izango dira inferentzia egiteko.

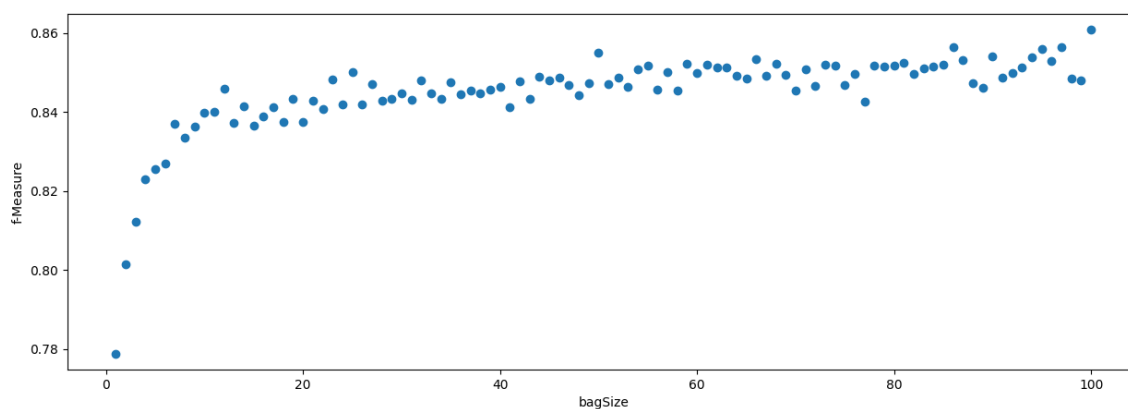
Behin parametro esanguratsuenen ekorketa egin dela, beste alde batera utzi diren parametroak ekortu dira (Irudia 11 eta Irudia 12).



Irudia 11: *numIterations* parametroa ekortzen

numIterations parametrarako lortu diren balio optimoak (Irudia 11) 100 balioaren hurbilenak izan dira, 102 lortu dugun optimoa izanik. Hausnarketa moduan, argi ikusten da zenbat eta iterazio, lagin edo *bootstrapping sample* gehiago izan, ez dela beti onuragarria. Lagin bakoitzak, kasu desberdinak planteatzen ditu baina momentu batetik aurrera, lagin gehiago izateak, lagungarri izateari uzten dio eta gero eta balio tarte handiagoa lortzen da.

Azkenik, *bagSizePercent* parametroaren ekorketa (Irudia 12) nahiko berehalakoa izan da.



Irudia 12: *bagSizePercent* parametroa ekortzen

Grafikoak adierazten duen bezala, lagin bakoitza instantzien kopuruaren %100-a izan behar du klase minoritarioaren *F-Measure*-a optimizatzeko.

Emaitzak eta ondorioak

Instantzien %5-erako lortu diren parametro optimoak:

- *maxDepth*: 109
- *numFeatures*: 297
- *numIterations*: 102
- *bagSizePercent*: 100

Balio hauek ia banan-banan ekortu ditugun arren, haien artean menpekotasuna dagoen ikusteko, batera ekortu dira beste fase batean baina ez dira aldaketa esanguratsuak lortu. Hala ere, konputazio ahalmen faltagatik, prozesu hau hobetzeko aukera dagoela kontuan izan behar da. Lortutako balioak, nahiko optimoak dira eta defektuzko *RandomForest* sailkatzailea (*baseline*) baino emaitza hobeak lortzeko ahalmena dauka.

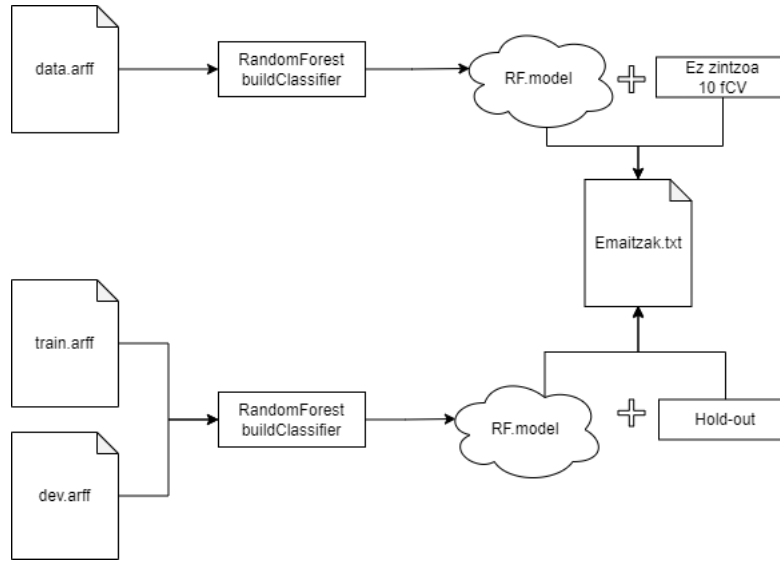
3.3.2 Eredua eta estimatutako kalitatea

Parametro optimoak ezarrita eta gainbegiratutako datu sorta osoarekin (*train* eta *dev*), eredua entrenatuko da geroago horren estimazioa lortzeko.

Ereduaren kalitatea estimatzeko hurrengo eskemak erabili dira:

- Ebaluazio ez-zintzoa (*resubstitution error*)
- 10-fold *Cross Validation*
- *Stratified Repeated Hold Out*

Lehen azaldu diren hiru eskema hauen inplementazioa, modelo optimoaren sortzearekin batera, eskema batean adierazi da (Irudia 13).



Irudia 13: Ebaluaketaren diagrama.

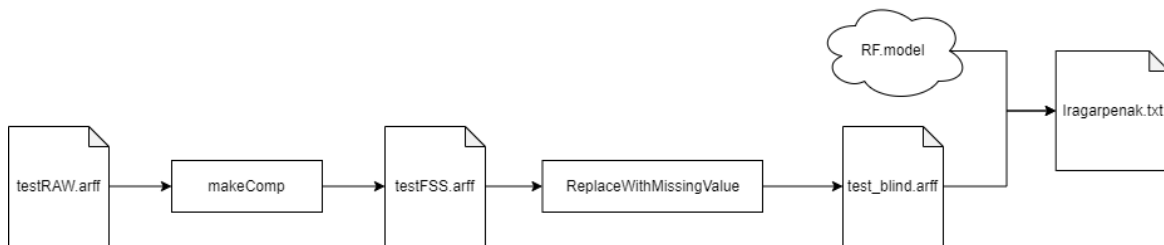
3.4 Sailkapena

Behin modeloa sortuta dagoela, instantzia berriak klasifikatu behar dira (testu ezberdinen klasearen balioa lortu), hori baita proiektuaren helburu nagusia.

Beraz, eta ikusteko errealitatean zer nolako fidagarritasuna duen modeloak, hasierako datueta-ko *testRAW.arff* instantzia multzoa eraldatuko da, modeloaren hiztegiarekin bateragarria izateko (*testFSS.arff* lortuz).

Test-aren instantziak sailkatuko dira *suicide* eta *non-suicide* klaseen artean eta errealitateko emaitzak lortutakoekin alderatuz aterako da zein den errealitatean lortu den modeloaren *accuracy*-a.

Lortzen diren iragarpen emaitzak dokumentu batean gordeko dira (Irudia 14).



Irudia 14: Iragarpenaren diagrama.

3.5 Emaizta esperimentalak

Erabili diren algoritmoetan emaitza (Taula 5, Taula 6) ezberdinak lortu dira. Alde batetik, *baseline* dago, honek kalitatearen behe borrea ezarriko du. Bestetik, *RandomForest* modeloaren emaitzak, atazarentzako optimoenak diren parametroekin entrenatu denez, emaitza onenak emango ditu.

	Class	Resubstitution error			10-folds Cross Validation		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
Baseline	+	1.000	1.000	1.000	0.881	0.839	0.859
	-	1.000	1.000	1.000	0.862	0.898	0.880
	W.Avg.	1.000	1.000	1.000	0.871	0.870	0.870
Model	+	1.000	1.000	1.000	0.890	0.874	0.882
	-	1.000	1.000	1.000	0.889	0.903	0.896
	W.Avg.	1.000	1.000	1.000	0.889	0.889	0.889

Taula 5: Ez-zintzoa eta 10-*fcV* ereduaren errendimendua

	Class	Stratified repeated hold out		
		Precision	Recall	F-Measure
Baseline	+	0.875	0.840	0.857
	-	0.863	0.894	0.878
	W.Avg.	0.869	0.868	0.868
Model	+	0.885	0.864	0.874
	-	0.881	0.899	0.890
	W.Avg.	0.883	0.883	0.883

Taula 6: *Stratified repeated hold out* errendimendua

Ebaluazio ez-zintzoan, oso errealista ez denez, kasu bietan kalitatearen goi borrea lortu da, kasu guztietan 1.000. 10 *folds Cross Validation*-ean, *baseline* eta modeloaren arteko ezberdintasunak ikus daitezke, *Precision*, *Recall* eta *F-Measure*en kasuan, gutxi gorabehera %1.6eko hobekuntza lortu da. *Stratified Repeated Hold Out*-ean ere hobekuntzak lortu dira, gutxi gorabehera %1.8koa.

Ikus daitekeen moduan azken bi ebaluazio eskemekin emaitza nahiko antzekoak lortu ditugu eta *baseline*-arekin konparatuta hobekuntzak ez dira oso nabariak. Emaizta hauek ahalmen konputazional murriztuaren ondorio dira, atributu eta instantzia asko bidean galdu baitira.

3.6 Emaizten diskusioa

Lehenik eta behin, esperimentalki lortutako emaitzak, oinarri teorikoak markatzen duenarekin bat datozela esan beharra dago. Izan ere, entrenatutako modeloarekin lortu izan diren emaitzak *baseline* emandako emaitzak baino hobeak dira. Bestalde, metodo ez-gainbegiratuak erabiliz, hau da, klasea kontuan hartu barik, egindako iragarpenetan %88 inguruko asmatze tasa lortu da. Hortaz, lortutako eredu sailkatzailea lagungarria izan daiteke tendentzia suizidak detektatzeko.

Emaizten konsistentzia aztertzeo iragarpenen prozesua hainbat alditan errepikatuko da, lortu-tako emaitzak kontrastatu ahal izateko. *CSV* ezberdinekin (inferentzia prozesuan erabili ez diren instantziak dituztenak) iragarpenak egin dira emaitzak jasoz (Taula 7).

Proba	Asmatze tasa (%)
1. Proba	99.00
2. Proba	86.80
3. Proba	78.37
4. Proba	92.00
5. Proba	93.23
6. Proba	85.55
7. Proba	85.71
8. Proba	88.88

Taula 7: Iragarpen prozesuaren emaitza ezberdinak

Taulatik ondoriozta daiteke iragarpenetan lortutako emaitzak ebaluazioak iragarritako emaitzen antzerakoak direla. Hau da, ebaluaketa atalean modeloak %88ko asmatze tasarekin ebaluatu da eta esperimentalki iragarpenak eginez batezbesteko asmatze tasa berdintsua lortu da. Hortaz, emaitzak kotsistenteak direla esan daiteke. Gainera, asmatutako zenbait instantzia sartu dira *CSV* batean eta instantzia bakoitzaren klasea iragarri da. Lortutako emaitzak zentzuzkoak dira (Irudia 15)

```

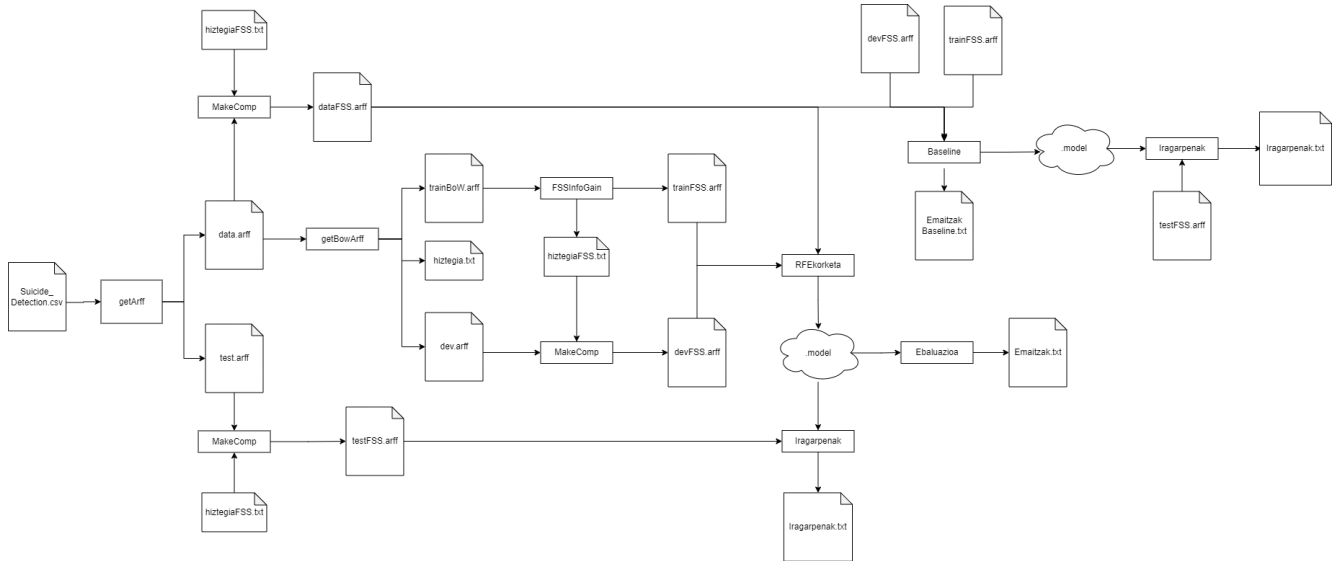
my whole body feels itchy and like its on fire, non-suicide
ahh ive always wanted to see rent love the soundtrack!!, non-suicide
this week is not going as i had hoped, non-suicide
im sad now Miss.Lilly *--* , non-suicide
"Put vacation photos online a few yrs ago.
PC crashed, and now
I forget the name of the site.", non-suicide
"Who am I? Someone that's afraid to let go, uh
You decide, if you're ever gonna let me know, yeah
Suicide, if you ever try to let go, uh|
I'm sad I know, yeah, I'm sad I know, yeah",
"I don't have social energy, i hope i was died", suicide
"I gave her everything
She took my heart and left me lonely
I think broken heart's contagious
I won't fix, I'd rather weep", non-suicide
"I really wanna die right now please let me go, im depressed", non-suicide
"Feeling suicidal right now i want to die, i want to kill myself i dont want
to live", suicide

```

Irudia 15: Asmatutako iragarpenen emaitzak

3.7 Exekuzio adibidea

Atal honetan exekuzioa era zuzenean burutzeko argibideak emango dira. Berez, modeloa sortzeko jarraitu izan den exekuzio fluxua diagrama batean adierazi da (Irudia 16). Hau da jarraitu beharreko prozesu osoa azkenengo iragarpenak izateko, baina bi modu daude exekuzioa burutzeko: script-en bidez edo exekutagarri solteen bidez.



Irudia 16: Exekuzioaren diagrama osoa

3.7.1 Script-en bidezko exekuzioa

Hiru exekutagarri sortu dira:

- Aurreprozesamendua.sh
- Inferentzia.sh
- IragarriCSV.sh

Script hauek exekutatzeke baldintza nagusien artean honako hauek daude:

- Script guztiak karpeta berdinarean barruan egongo dira (adibidez, 'WEKA' izeneko karpeta batean)
- Karpeta horren barruan beste lau karpeta egon beharko dira izen hauekin: 'Fitxategiak', 'Emaitzak', 'JAR', 'CSV'
- 'CSV'-aren barruan, bi fitxategi egongo dira 'Suicide_Detection.csv' (hau izan da erabili izan diren datuak modeloa entrenatzeko; esteka honetan aurki daitezke: datuak) eta 'Predictions.csv' (iragartzeko instantzia ezezagunak dituen fitxategia; formato honekin instantzia bakoitza: *"testua"*, *NaN (enter)* edo *"testua"*, *suicide/non-suicide (enter)*)

- ‘JAR’ karpetan, emandako .jar guztiak egongo dira (*script*-etan exekutatu direnak)
- Fitxategiak karpetan sortutako *ARFF*, hiztegiak eta .model guztiak egongo dira (emandako modelo optimoa barne)
- Emaitzak karpetan egongo dira *baseline*, ebaluazio eta iragarpenetatik ateratako emaitzak
- Script-etan adierazten den bezala *JAVA* erabiliko da hauek exekutatzeko, `/usr/local/java/jdk-19.0.2/bin/java` karpetaren barruan aurkitu behar den *JDK jdk-19.0.2* bertsioa hain zuzen ere. **Hau aldatu nahi izatekotan (bertsioa edo karpeta) *script*-en barruan adierazi beharko da.**

Lehenengo bi *script*-ak (aurreprozesamendua eta inferentzia), ordena horretan exekutatu behar dira, izan ere, inferentzia burutzeko aurreprozesamenduko fitxategiak behar dira. Hauek exekutatzeko soilik ‘Suicide_Detection.csv’ adierazitako karpetan izatearekin nahiko izango da. Behin biak exekutatuta ‘RF.model’ modelo bat egongo da ‘Fitxategiak’ karpetan eta emaitzetan hiru dokumentu egongo dira ebaluazio, *baseline* eta hasierako fitxategitik soberan dauden instantzietatik sortu izan den multzo baten iragarpenak (asmatze tasarekin amaieran) egongo dira.

Bestetik, ‘IragarriCSV.sh’ exekutatzeko, aipatutako ‘Predictions.csv’ eta adierazten den modeloa (‘ModeloOptimo.model’) ‘Fitxategiak’ karpetan egotea beharrezkoa da (beste ‘.model’ bat frogatu nahi izatekotan izena aldatu honi edo *script*-ean dagoen izena aldatu).

Modu honetan gauzak exekutatzerakoan, ez da posible izango pausu solte bat errepikatzea gaizki atera bada exekuzioaren amaieran, gainera *JAVA*-ko *JDK* kokalekua aldatzea zaila izan daiteke (edo beste aukera guztiak). Baina erabilgarria suerta daiteke prozesu luzeak exekutatzen uzteko zerbitzari batean.

3.7.2 Exekutagarrien bidezko exekuzioa

Pausu bat modu soltean exekutatzeko aukera ere badago: ‘.jar’-ak erabiltzen. Hauek, lehen aipatutako ‘JAR’ karpetaren barruan egonda, hurrengo baldintzak bete behar dute haien exekuziorako:

- Gomendatzen da lehen aipatu den ‘WEKA’ izeneko karpetatik exekutatzeko exekutagarriak, script barruko estruktura bera jarraitu ahal izateko.
- Berez, komandoaren hasieran “*java -jar*” idaztarekin nahiko izan ahal da, baina *JDK* aurkitu ez duelaren errore bat eman ezker, *JDK*-ko java dagoen direktoriorako path-a idatzi honen aurretik (script-etan agertzen den moduan).
- “*java -jar*” ostean “*-add-opens java.base/java.lang=ALL-UNNAMED*” jartzea gomendatzen da, *WEKA*-ko liburutegiarekin errorerik ez izateko.
- Horren ostean, ‘.jar’ exekutagarriaren path-a eta honi dagozkion argumentuak jarriko dira.

Dauden *JAR* guztien erabileraren azalpena ‘JAR’ karpetaren barruan dagoen ‘README.txt’ fitxategian dator. Bertan xehetasun handiagoarekin azaltzen da zer egiten duen bakoitza. Dena den, hemen exekuzio adibide bana jarriko da:

- **getArff.jar**

Exekutagarri honen bidez ‘Suicide_Detection.csv’ fitxategia *ARFF* fitxategi batean bihurtuko du adierazitako ehunekoa eta soberan dauden instantzietatik test.arff bat sortuko du.

```
java -jar path/to/getArff.jar path/to/Suicide_Detection.csv path/to/irteerako/dataRAW.arff  
ehunekoa path/to/irteerako/testRAW.arff
```

- **arff2bow.jar**

Honen bidez, aurretik sortutako *ARFF*-a *Bag of Words* batean eraldatuko du (0/1 aukerak ‘README’ barruan azalduta datoz).

```
java -jar path/to/arff2bow.jar path/to/dataRAW.arff ‘0/1’ ‘0/1’ path/to/irteerako/hiztegia.txt  
path/to/irteerako/trainBOW.arff path/to/irteerako/devRAW.arff
```

- **fssInfoGain.jar**

Aurrekoan lortutako *BoW*-a atributuen selekzio baten bidez filtratuko da hemen (2000 atributuekin geratuz).

```
java -jar path/to/fssInfoGain.jar path/to/trainBOW.arff path/to/irteerako/trainFSS.arf path/-  
to/hiztegia.txt path/to/irteerako/hiztegiaFSS.txt
```

- **MakeComp.jar**

Aurreko metodoan sortutako hiztegiarekin konpatibleak egingo ditu sartzen diren *ARFF* berriak (0/1 aukerak ‘README’ barruan azalduta datoz).

```
java -jar path/to/MakeComp.jar path/to/inputData.arff path/to/irteerako/dataFSS.arff ‘0/1’  
‘0/1’ path/to/hiztegiaFSS.txt
```

- **ParametroEkorketa.jar**

Aurreprozesamendu osoa burutu ostean sortutako fitxeroetatik modelo berri bat entrenatzeko parametro ekorketa burutuko du *RandomForest* klasifikadorearentzat (prozesu oso luzea da, beraz, zerbitzari batean exekutatzeari gomendatzen da).

```
java -jar path/to/ParametroEkorketa.jar path/to/trainFSS.arff path/to/devFSS.arff path/to/-  
dataFSS.arff path/to/irteerako/ParametroEkorketaEmaitzak.txt
```

- **Baseline.jar**

Defektuzko balioak dituen *RandomForest* batekin *FSS* prozesutik ateratako datuen ebaluazioa burutuko da.

```
java -jar path/to/Baseline.jar path/to/trainFSS.arff path/to/devFSS.arff path/to/dataFSS.arff  
path/to/irteerako/EvaluationBaseline.txt
```

- **Ebaluazioa.jar**

Parametro ekorketan lortutako parametro optimoekin ateratako modeloaren ebaluazioa burutuko da *FSS* prozesutik ateratako datuekin. Frogatutako parametro optimoak hurrengoak dira, hurrenez hurren: 297, 102, 100 eta 109.

```
java -jar path/to/ParametroEkorketa.jar path/to/trainFSS.arff path/to/devFSS.arff path/to/-  
dataFSS.arff 'NumFeatures' 'NumIterations' 'BagSizePercent' 'MaxDepth' path/to/irteerako/E-  
valuationAlgorithm.txt
```

- **InputIragarpenak.jar**

Lehen aipatutako 'Predictions.csv' *blind* instantzien fitxategia aurreprozesu osotik pasatuko du (*ARFF*-rako bihurtuta eta 'MakeCompatible'-tik).

```
java -jar path/to/InputIragarpenak.jar path/to/Predictions.csv path/to/testRAW.arff path/-  
to/testFSS.arff path/to/irteerako/hiztegiaFSS.txt
```

- **Iragarpenak.jar**

Adierazitako modeloa erabiliz adierazitako *test* multzoaren iragarpenak egingo dira hemen. Test-a *blind* zein gainbegiratua izan ahal da.

```
java -jar path/to/Iragarpenak.jar path/to/Modeloa.model path/to/testFSS.arff path/to/devFSS.arff  
path/to/irteerako/TestPredictions.txt
```

4 Ondorioak eta etorkizunerako lana

Interneteko mezuetatik abiatuta, mezuaren idazlea bere buruaz beste egingo duen edo ez iragartzea lortu da proiektu honetan. *RandomForest* algoritmoa erabili da horretarako, sailkatzaile gainbegiratu bat erabaki-zuhaitz asko sortzen dituen hauetako bakoitzetik ateratako emaitzak konbinatuz, eredu sendoagoa lortzen duena iragarpenak egiteko.

Uste da garatutako lana nahiko eraginkorra dela, izan ere, aurretik aipatutako emaitzetan ikusi daiteke nola defektuzko balioak dituen *RandomForest* sailkatzaile batekin konparatuz, eraikitako modeloa emaitza hobeak lortzen dituen datu berdinentzat. Noski, lanak bere ahuleziak ditu, baina baita sendotasunak ere.

Ahuleziei dagokionez, garatutako lanak, lortutako emaitzak ikusita, heldutasun nahikoa duela uste da. Halere, hobekuntza bide batzuk aurkitu dira.

Alde batetik, hasierako *.arff*-a lortzeko metodoa ezin da beste *CSV* motako fitxategiei aplikatu, izan ere, hasieran jasotako *CSV*-aren instantziak nahiko zikinak ziren (*enter* ugari eta instantzien hasierak zein amaierak gaizki definituak zeuden). Hori dela eta, metodoa eskuz egin da jasotako datuei egokitua. Hortaz, garatutako *CSV-ARFF* bihurgailua ez da eskalagarria, soilik kasu honetarako aplikagarria baita. Dena den, aldaketa txiki batzuk eginez, beste *CSV* mota batentzako erabilgarria bihur daiteke, oinarritzak egitura antzekoa baita.

Bestetik, nahiz eta azkenengo emaitzak oso onak izan, egia da nahasmen matrizeak begiratuta, *FPR* (*False Positive Rate*) eta *FNR* (*False Negative Rate*) nahiko berdinak izanda, *FPR* bestea baino txikiagoa dela (Irudia 17). Honek esan nahi du suizidio posible asko baztertuko direla selekzio prozesuan. Hau aldatzea *FNR* txikiago bat lortuz hobekuntza arlo posible bat izan zitekeen, errore mota denak ez dute inpaktu berdina programaren aplikazio edo erabilpen fasean. Suizidioen kontuan askoz larriagoa da *FNR* altu bat izatea *FPR* altu bat izatea baino.

```

=== Confusion Matrix ===
      a      b  <-- classified as
15732  2302 |      a = suicide
 2032 18103 |      b = non-suicide

```

Irudia 17: Ebaluazioko nahasmen matrizea.

Sendotasunei dagokionez, esan bezala, proiektuak heldutasun maila nahikoa duela kontsideratzen da. Batez ere, jasotako datu gordinei probetxurik handiena ateratzea lortu delako. Eskuragarri zegoen ahalmen konputazionala kontuan harturik, instantzia kopuru handiekin lan egin da, emaitza hobeak lortuko zirelakoan.

Hasierako *.arff*-a lortzeko metodoa kenduta, softwarea malgua da, izan ere, erabiltzaileak ezarri dezake zein den erabiliko diren datu multzoak, zein errepresentazio bektorial erabili nahi den, *Sparse/NonSparse...* Modu honetan, testu meatzaritzako antzeko atazetan erabiltzeko aukera ematen du.

Lanarekin berriro hastekotan, uste da batez ere implementatzen hasi aurretik proiektuaren diseinua oso argi egon beharko zela, azken astera arte honi buruzko zalantzak agertu baitira. Horrez gain, zenbait kontzeptu hasieratik ezartzea ere garrantzitsua izango litzateke, atributu kopurua hasiera batean, atributuen hautapena...

5 Balorazio subjektiboa

Zeregin honen helburu nagusia datu meatzaritzako ataza batean jarraitu beharreko prozedura ezagutzea eta praktikan jartzea izan da. Datu gordinekin lan egin behar izateak aurreprozesamenduaren garrantziaz konturatzea ekarri du. Gainera, nahiko ezezaguna den gai baten gainean lan egitea interes handia pizten du, lagungarria izan daitekeela pentsatuz.

Une honetan, proiektua bukatuta dagoelarik, helburuak bete direla esan daiteke, adierazitako modulu guztiak egoki funtzionatzen baitute. Alde batetik, datuen karga eta garbiketa burutzea lortu da eta Wekarentzat interpretagarriak diren datuak lortu dira. Bestetik, datu horiekin inferentzia prozesua egin eta sailkatzaile optimoa lortu da parametro ekorketa eginez. Gainera, *baseline* eta entrenatutako sailkatzailearen kalitatea estimatu da ebaluazio prozesu baten bitartez. Bukatzeko, iragarpenak egiteko funtzionalitatea garatu da, hasierako helburu nagusiari irtenbidea emanez.

Ataza honen zailtasun handienetarikoa esparru honetan izandako praktika falta da. Orain arte honelako teknikak ez ditugu aplikatu eta beraz softwarearen garapenak arazo berriak sortarazi ditu. Arazo berriei aurre egiteak denbora aldetik eragina izan du. Inbertitutako denbora, moduluz modulu, taula (Taula 8) batean adierazi da.

Ataza	Inbertitutako denbora
Ikasketa denbora	22h
Bilaketa bibliografikoa	8h
Software diseinua	16h
Software inplementazioa	200h
Dokumentazioa	30h

Taula 8: Denbora banaketa taula

Beste alde batetik, proiektu hau era modularrean garatu izan da 4 pertsonen arteko elkarlanean. Lagungarria izan da taldeka lan egitea ataza hau burutzeko. Gauzak horrela, garatu beharreko modulu ezberdin bakoitzeko arduradun bat edo batzuk zehaztu dira, hala ere, taldekide orok besteek egiten zutenaren berri zuten eta indibidualki landutako gauza denak geroago taldekideei azaldu zaie. Ondorioz, kasu honetan izugarri positiboa izan da talde lanean aritzea.

Praktika honetan zehar motibazio handiena ‘errealitateko’ datuekin lan egitea izan da, gainera, inpaktu sozial handia duen gai baten gainean. Orain arte garatutako zereginetik oso ezberdina izan da esperientzia eta kontzeptu dezente ikasi dira bai diseinu eta baita inplementazio ataletan zehar. Azkenik, *data mining* esparruko terminologia eta hainbat kontzeptu ulertzea eta barneratzea oso positiboki baloratzen da.

6 Bibliografia

1. Suicide and Depression Detection dataset (Kaggle).
2. Wekako wikia: <https://waikato.github.io/weka-wiki/>
3. Weka Javadoc: <https://weka.sourceforge.io/doc.dev/overview-summary.html>
4. RANDOM FORESTS, Leo Breiman (Statistics Department, University of California)
5. Analysis of a Random Forests Model, Gerard Biau (LSTA & LPMA, Université Pierre et Marie Curie – Paris VI)
6. Aplicación de algoritmos Random Forest y XGBoost, Javier Jesús Espinosa-Zúñiga
7. Tutorial para un clasificador basado en bosques aleatorios: cómo utilizar algoritmos basados en árboles para el aprendizaje automático
8. Bagging in Machine Learning: Step to Perform And Its Advantages
9. Bootstrap aggregating, Wikipedia
10. Random Forest Algorithm Clearly Explained!, Normalized Nerd
11. Bootstrap aggregating, Wikipedia