Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

PRÀCTICA 2: DETECCIÓ D'OPINIONS

Processament del Llenguatge Humà

Grau en intel·ligència artificial

Mikel Lecumberri Arteta i Elies Aragonès Larrañaga

25/03/2025

Índice

1. Introducció

Les *reviews* s'han convertit en un factor molt clau per a l'èxit dels negocis, ja que serveixen com a un indicador de qualitat. És per aquest motiu que és molt important per les empreses poder analitzar-les i corregir els errors. Això passa també sovintment a l'entreteniment, com ara a pel·lícules i llibres. És per aquest motiu que hi ha eines, com la detecció d'opinions, que permet agilitzar aquest error.

Per tant, el nostre objectiu en aquest treball ha sigut provar diferents models de *machine learning*, tant de supervisats com de no supervisats, per a fer prediccions sobre si una opinió és positiva o negativa.

Aquest procés inclourà un preprocessament d'una base de dades sobre opinions de pel·lícules, el tuneig dels hiperparàmetres dels models per tal de trobar el millor i les prediccions finals amb el conjunt de test. També cal mencionar que el que es farà en aquest informe serà repassar el codi d'una manera més general i conceptual, i és al *notebook* on aprofundirem més a explicar les nostres decisions i la implementació de tots els passos.

2. Preprocessat

El primer pas va consistir a fer la funció que serviria per fer el preprocessament de cadascuna de les opinions. Això és molt important, pel fet que ens permetrà tenir totes les paraules en el mateix format i eliminarem soroll. Per fer això, utilitzarem funcions de *nltk*, expressions regulars, *unidecode* i mètodes del mateix *python*.

El primer pas consistirà a passar tot el text a minúscules. Això és necessari perquè, si no, pot detectar una mateixa paraula de dues maneres diferents només pel fet que una sigui començament de frase i comenci amb majúscules, el qual seria un problema. Pel mateix motiu també vam utilitzar el mètode *unidecode()* per treure els accents.

A continuació, vam fer servir expressions regulars per eliminar els dígits de les opinions, ja que vam considerar que no aportarien informació important i afegirien soroll. Tot seguit, vam tokenitzar el text i vam eliminar els elements que no fossin alfanumèrics, és a dir, signes de puntuació, pel fet que tampoc era rellevant. Finalment, ens vam quedar només amb el lema de cada paraula, perquè és el que de veritat conté la informació d'aquesta.

També cal mencionar que, tot i que en un primer moment es va implementar també la funció per eliminar les *stopwords*, pel fet que no aporten informació rellevant, es va treure perquè les funcions de *CountVectorizer* i *TfidfVectorizer* ja ho fan elles mateixes si posem *stop_words* com a paràmetre juntament amb l'idioma amb què estem treballant.

Per la segona part de la pràctica el preprocessament era molt semblant, quasi idèntic. Tot i això, es van provar diferents estratègies per abastar els límits del model. Es va afegir una eliminació dels noms propis i es va proposar, com a la primera part l'eliminació de *stop words*. No obstant aquesta última tècnica no va donar com a resultat millores en el model quant a eficàcia i resultats, però sí que afegia un temps considerable al model, per tant, es va desestimar. Es va separar en els mateixos conjunts de *train* i *test* per tal de comparar els resultats supervisats i no supervisats.

3. Lectura de dades i particions

Per tal d'obtenir les dades vam descarregar-nos una base de dades de *nltk* anomenada *movies_reviews*. Després, vam seleccionar les opinions positives seleccionant els arxius del corpus que tenien la ID *pos* i les vam guardar a una variable. Posteriorment, vam fer exactament el mateix amb les opinions negatives, aquesta vegada filtrant per l'etiqueta *neg*. Totes dues tenien exactament la mateixa mida: 1000 opinions.

El següent pas va consistir a fer les particions. En el nostre cas vam decidir fer únicament train i test, pel fet que els paràmetres es buscarien amb validació creuada, així que no era necessari crear un conjunt de validació. Pel que fa als percentatges, vam decidir assignar un 80 % al train, que equival a 1600 opinions, i un 20 % al test, que equival a 400. Vam escollir aquests percentatges perquè vam trobar que ens permetrien tenir un bon conjunt d'entrenament i alhora tenir prou elements per fer la validació del model final.

Un cop sabíem ja els percentatges, vam seleccionar les 800 primeres opinions positives i les 800 primeres de les negatives i les vam concatenar en una nova variable que seria el train, i el mateix amb les 200 restants dels dos grups pel test. Un cop fet això, vam llegir totes les opinions utilitzant el mètode *.words()*, ja que fins ara el que teníem eren els identificadors de cada opinió. L'últim pas va consistir a aplicar la funció de preprocessat que hem comentat anteriorment a cadascuna de les opinions, i d'aquesta manera teníem ja *train_x* i *test_x* preparats.

Finalment, només queda ja fer les llistes amb les etiquetes tant del train com del test. En aquest cas, com les opinions estan per ordre, l'únic que haurem de fer serà concatenar una llista d'uns amb una llista de zeros de la mida de la partició. És a dir, *train_y* serà una llista amb 800 uns seguits de 800 zeros, i *test_y* seran 200 uns seguits de 200 zeros.

4. Models supervisats

El que farem en aquesta secció serà provar diferents models per tal de veure quin és el que dona millors resultats. Per fer les prediccions utilitzarem un vectoritzador que entrenarem i crearà una taula de freqüències de les paraules a cada document. A continuació, es farà una matriu també per al text que volem predir, però amb el vocabulari amb el qual l'hem entrenat i un model de *Machine Learning* és l'encarregat de fer les prediccions.

Per tal de fer les vectoritzacions existeixen principalment dos models: *CountVectorizer()* i *TfidfVectorizer()*. En aquest treball els provarem tots dos per tal de veure quin obté millors resultats. En aquest cas, el primer compta les freqüències de cada paraula, mentre que el segon utilitza un càlcul més complex que redueix el pes de les paraules més comunes, reduint així el soroll. Podem veure com ho calcula exactament a continuació:

$$\text{TF}(t, d) = \frac{\text{Nombre de vegades que el terme } t \text{ apareix al document } d}{\text{Total de termes al document } d}$$

$$\text{IDF}(t) = \log\left(\frac{\text{Nombre total de documents}}{\text{Nombre de documents que contenen el terme } t}\right)$$

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

4.1. Prova de models

4.1.1. Enfocament

El primer pas va consistir a provar un gran nombre de models per tal de veure quin és el que dona millor *accuraccy*. Ara bé, en aquest cas el vectoritzador també s'haurà de tunejar, concretament els paràmetres de *max_df* i *min_df*, que serveixen per definir la

freqüència màxima i mínima de les paraules per tal que s'utilitzin a la vectorització. Llavors, l'ideal en aquest cas seria tunejar els dos models a la vegada, és a dir, provar moltes combinacions de paràmetres del model de predicció, i per cada una d'aquestes combinacions provar moltes combinacions de paràmetres de tots dos vectoritzadors, i això per a cada model.

Ara bé, per fer aquesta cerca tan exhaustiva que hem comentat serien necessaris molts recursos i molt de temps. És per aquest motiu que vam decidir fer una versió una mica més senzilla: per a cada model vam seleccionar l'hiperparàmetre més important i vam provar tres valors diferents d'aquest, i per a cada un d'aquests provar tots dos vectoritzadors amb diferents paràmetres. Tot i que aquest no serà el mètode més eficient ens permetrà tenir una visió general.

4.1.2. Codi

Passant ara ja al codi el primer que va fer va ser una funció que retorna un diccionari amb tots els paràmetres a provar, tant pel vectoritzador com pel model de predicció. Per altra banda, la funció on es busquen els paràmetres de cada model és la de *gridsearch*. El primer que es fa és crear el *pipeline*, que servirà per aplicar el vectoritzador i, seguidament, el model. Després, es crida la funció que hem comentat anteriorment per obtenir els paràmetres.

Quan ja tenim aquestes dades podem crear el *GridSearchCV* utilitzant 5 folds i basant-nos en la accuracy com a mètrica. Cal mencionar que, gràcies al pipeline, la vectorització es farà després de la divisió per folds i no abans, el qual seria un error.

Finalment, el que farem serà cridar a una funció anomenada *trobar_milloe_model()*, que el que farà serà cridar la funció de *gridsearch* per a cada model, que en aquest cas hem decidit utilitzar: regressió logística, Naïve Bayes, Random Forest, KNN, gradient boosting i SVM.

4.1.3. Resultats

A continuació, el que farem en aquest apartat serà analitzar els resultats obtinguts després d'executar les funcions que hem comentat anteriorment. Per fer-ho, ens fixarem principalment en la matriu de confusió i l'accuracy, tot i que també mencionarem altra informació que ens ofereix aquesta funció.

En primer lloc, comentarem la regressió logística, la qual va obtenir que els millors paràmetres van ser una C de 10 i una vectorització amb Tfidf amb 0.6 de max_df i 0.0 de *min_df*. L'accuracy obtinguda va ser de 0.839 i a la figura 1 podem veure com tant per les opinions negatives com les positives el nombre de prediccions correctes i incorrectes és el mateix.

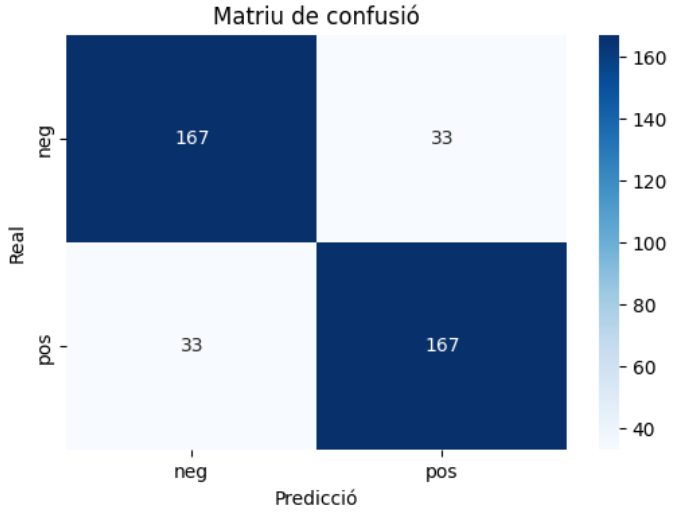


Figura 1: Matriu de confusió de la regressió logística

El següent model que vam provar va ser Naïve Bayes, el qual combina el teorema de Bayes amb la suposició de Naïve (que assumeix que totes les característiques són independents entre si) per fer les prediccions. En aquest cas, l'hiperparàmetre que vam modificar va ser *alpha*, i ens va donar que el valor òptim era 1. Per altra banda, el millor vectorizador va ser Tfidf de nou amb 0.8 de max_df i 0.0 de min_df.

En aquest cas l'accuracy va ser de 0.817 i la matriu de confusió és la que veiem a la figura 2. Pel que fa a aquest model, sí que veiem diferències entre les prediccions, ja que les opinions negatives, amb 35 errors, van ser millor classificades que no pas les positives, amb 53. Això ho podem veure també amb el *classifiaction report* (es pot veure al *notebook*), pel fet que les opinions negatives tenen un recall de 0.82 davant d'un 0.73 de les positives.

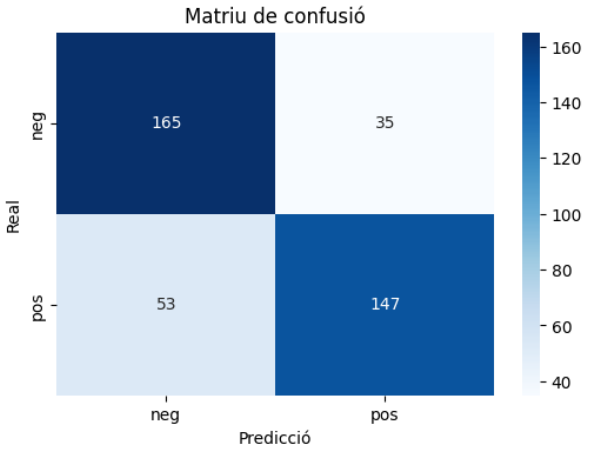


Figura 2: Matriu de confusió de Naïve Bayes

El pròxim model que vam provar va ser el Random Forest, el qual utilitza molts decision trees i combina les prediccions. Vam obtenir que el millor valor per a *n_estimators* és 200 i que la millor vectorització és amb CountVectroizer amb 0.8 de max_df i 0.0 de min_df. L'accuracy del model va ser de 0.826 i la matriu de confusió la podem trobar a la figura 3. En aquest cas observem com de nou les opinions negatives han sigut millor predites, però aquesta vegada amb una mica menys de diferència. El recall en aquest cas és de 0.85 per les negatives i 0.8 per les positives.

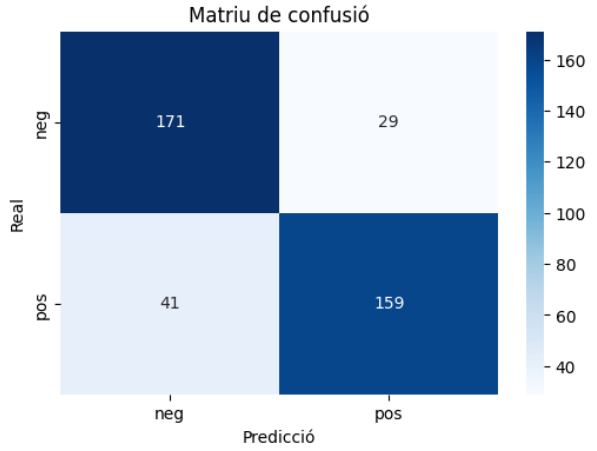


Figura 3: Matriu de confusió del random forest

A continuació, veurem els resultats obtinguts per a KNN, que va obtenir els millors resultats amb un nombre de veïns de 3 i una vectorització amb Tfidf amb 0.8 de max_df i 0.0 de min_df. L'accuracy d'aquest model va ser la més baixa amb diferència, pel fet que va ser de 0.684. Si observem la matriu de confusió de la figura 4 podem veure com, efectivament, el nombre de prediccions equivocades és molt major al que havíem vist fins ara.

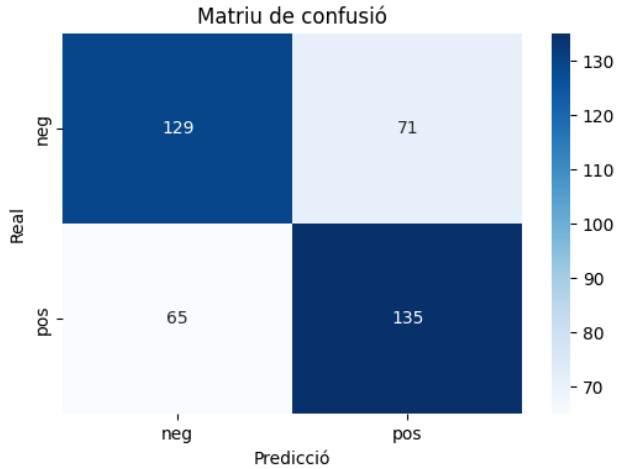


Figura 4: Matriu de confusió de KNN

El següent model que vam provar va ser el de *gradient boosting*, el qual va obtenir els seus millors resultats amb un *learning rate* de 0.1 i el CountVectorizer amb 0.8 de max_df i 0.0 de min_df. L'accuracy obtinguda també va ser una mica menor a les que havíem vist en els primers models, sent de 0.789. Observant la seva matriu de confusió de la figura 5 podem veure que segueix el mateix patró que les anteriors: les opinions negatives estan millor classificades que les positives.

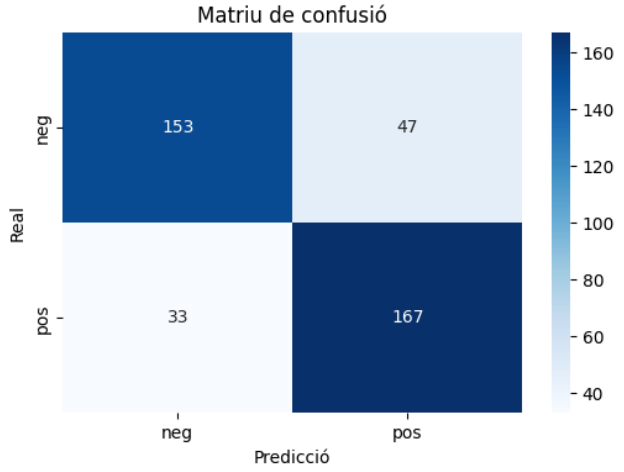


Figura 5: Matriu de confusió de Gradient Boosting

Finalment, l'últim model que provarem serà el SVM amb una C igual a 10 i el Tfidf com a vectorització, amb 0.6 de max_df i 0.0 de min_df. L'accuracy obtinguda va ser de 0.830 i, tal com es pot veure a la matriu de confusió de la figura 6, en aquest cas la classificació de les opinions positives va ser lleugerament millor.

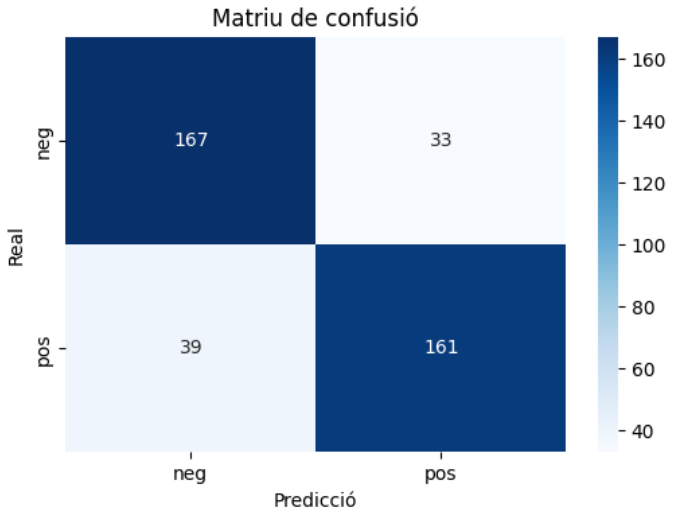


Figura 6: Matriu de confusió de SVM

Per acabar, un cop analitzats tots els models, ara tocarà decidir quin ha sigut el que ha obtingut millors resultats. Tal com podem veure a la figura 7, menys el KNN i el gradient boosting, tots han obtingut accuracies bastant similars. En aquest cas, però, hem decidit quedar-nos amb la regressió logística pels següents motius: és la que ha obtingut una accuracy més alta, classifica igual de bé les dues classes i té un temps d'execució baix. Aquest últim punt és el que ens ha fet triar la regressió logística per sobre del SVM, ja que aquest segon tenia també una accuracy alta i les prediccions equilibrades.

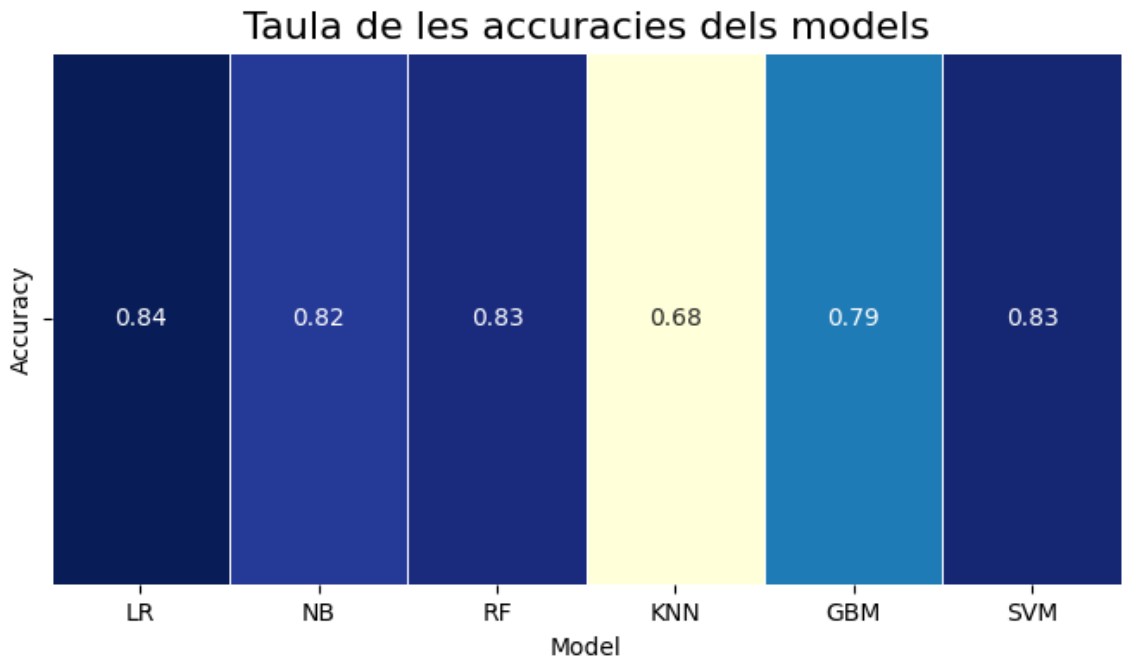


Figura 7: Taula amb les accuracies de tots els models

4.2. Millor model

Un cop ja sabem que la regressió logística serà el nostre millor model, ara ja podrem tunejar més hiperparàmetres per tal de donar amb la millor combinació possible. En aquest cas el que farem serà molt similar al que hem fet anteriorment: cridarem a la nostra funció de *gridsearch()* per obtenir el millor model possible. L'única diferència és que aquesta vegada canviarem la llista de *params_grid* i provarem també diferents *solvers* i *max_iter*. També farem una petita modificació als valors de min_df, pel fet que ens hem adonat que sempre donava que 0.0 era el millor perquè segurament la resta de valors eren massa grans.

Després de provar un gran nombre de combinacions de paràmetres, tant de la regressió logística com del vectoritzador, vam obtenir l'accuracy més alta de 0.8412 que, tot i no ser molt més alta que la que teníem inicialment, sí que ha significat una millora. Aquest

nou model té els següents paràmetres:

- C = 10.0
- max_iter = 500
- solver = saga
- vectoritzador = TfidfVectorizer
- max_df = 1.0
- min_df = 0.02

El primer que veiem és com, efectivament, el canvi que vam fer en els valors de min_df van ser positius, ja que, en aquest cas, el millor model n'ha tingut 0.02 en aquest paràmetre. El que fa això és que no es tingui en consideració paraules que apareixen molt poc i que, per tant, poden afegir soroll. Tot i ser un canvi petit, té un impacte molt gran en la vectorització, pel fet que, en aquest cas, el vector passa de tenir mida 32.000 a 2300.

4.2.1. Resultats

Un cop ja sabem els paràmetres del nostre model final, ja podem passar a fer les prediccions amb el nostre conjunt de test. En aquest cas, vam crear una nova funció, la qual obté el model final com a paràmetre, fa les prediccions i ofereix diferents mètriques per analitzar els resultats.

L'accuracy obtinguda va ser lleugerament inferior a la que teniem durant el tuneig, de 0.8225, el qual podria voler dir que va fer una mica d'*overfitting*. El primer que veiem és la matriu de confusió a la figura 8. El primer que observem és que les opinions positives van ser lleugerament pitjor predites, amb 38 errors en comparació dels 33 de les opinions negatives.

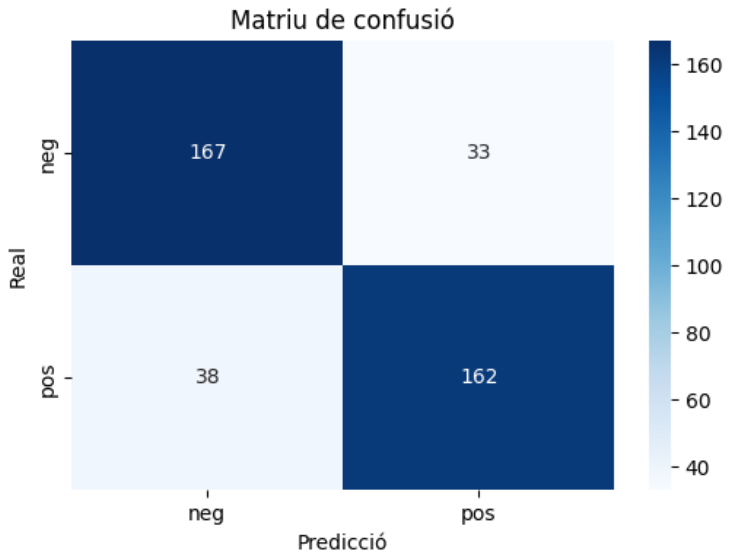


Figura 8: Matiu de confusió del model final

Això mateix ho podem veure també al *classification report* de la figura 18, on observem una taula amb els resultats mesurats en diferents mètriques. Concretament, ens podem fixaren el recall, el qual es calcula dividint els positius correctament predits entre el total de positius reals. En aquest cas, el de les negatives és lleugerament superior, el qual era esperable després de veure la matriu de confusió.

	precision	recall	f1-score	support
neg	0.81	0.83	0.82	200
pos	0.83	0.81	0.82	200
accuracy			0.82	400
macro avg	0.82	0.82	0.82	400
weighted avg	0.82	0.82	0.82	400

Figura 9: Classification report del model final

Finalment, l'últim que analitzarem serà la corba ROC del model, la qual podem observar a la següent figura. Pel que fa a aquest tipus de corbes, el punt òptim acostuma a ser el més pròxim a la cantonada superior esquerra, ja que és el punt on es maximitzen els positius correctes i es minimitzen els falsos positius. Ara bé, aquest llindar pot canviar depenent de la precisió que requereixi l'ànalisi.

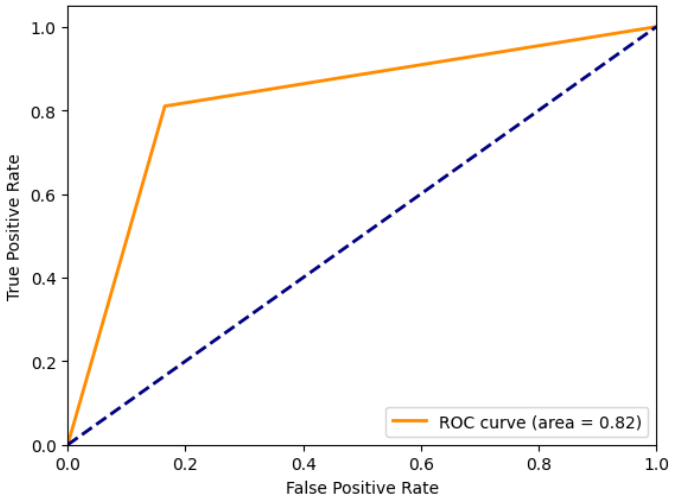


Figura 10: Corba ROC del model final

4.2.2. Anàlisi d'errors

Finalment, l'últim pas va consistir a analitzar totes aquelles opinions que van ser classificades de manera errònia per tal d'intentar identificar patrons. El primer pas va consistir a recórrer les prediccions i anar imprimint totes aquelles opinions que estiguessin mal predites.

Tot i que, en un inici, el nostre objectiu era anàlisi cadascuna de les opinions mal predites per separat, ens vam adonar que això era molt complicat per la longitud d'aquestes. És per això que vam intentar obtenir la màxima informació possible a partir de càlculs i comprovacions.

El primer que vam detectar és que les opinions tenien una mida molt gran, major a la qual nosaltres havíem imaginat en un inici. El nostre primer pensament va ser que potser la longitud de les opinions podria tenir relació amb la qualitat de les prediccions. És per això que vam calcular la mitjana de la longitud de les opinions correctament predites i la de les incorrectament predites.

Longitud de les prediccions correctes	Longitud de les prediccions incorrectes
679.927	611.732

Tabla 1: Longitud de les prediccions correctes i incorrectes

Tal com es veu a la taula 1, la diferència entre la mitjana de les longituds de les prediccions correctes i les incorrectes és molt notable, de quasi 70 paraules. Això ens demostra que és possible que les opinions més curtes aporten menys informació i, per això, tenen més probabilitat de ser predites de manera incorrecta.

Ara bé, analitzant les opinions podem veure com moltes d'elles contenen, també, un resum de la pel·lícula. El que provoca això és que hi hagin més paraules no polaritzades, el qual l'únic que fa és afegir soroll. És per això que vam decidir comptar també la mitjana d'adjectius per opinió de les correctament classificades i de les incorrectament classificades. Vam decidir fer això perquè els adjectius són els que acostumen a aportar aquesta polarització.

Adjectius de les prediccions correctes	Adjectius de les prediccions incorrectes
39.699	33.718

Tabla 2: Mitjana d'adjectius per opinió de les prediccions correctes i incorrectes

Tal com es pot veure a la taula 2, efectivament, les prediccions correctes acostumen a tenir més adjectius que les incorrectes. Això ens pot fer pensar que, tal com hem dit abans, el fet de tenir més adjectius aporta més polarització. Finalment, també vam trobar interessant veure quins són els adjectius que més es repeteixen a les prediccions mal predites.

Real positiva | Predicció negativa		
	Paraula	Freqüència
50	good	50
71	little	32
88	new	32
92	original	29
9	best	20
11	big	20
51	great	18
102	real	16
7	bad	16
56	high	14

(a) Opinions positives mal predites

Real negativa | Predicció positiva		
	Paraula	Freqüència
35	good	27
78	new	24
60	little	19
36	great	19
80	original	18
10	best	18
136	young	17
41	hard	16
12	big	13
8	bad	13

(b) Opinions negatives mal predites

En aquest cas, tal com es pot observar a la figura, els adjectius més repetits per les prediccions positives i negatives són molt similars. El més repetit, en tots dos casos, és *good*, però observem com apareix quasi el doble de vegades a les opinions positives mal predites, el qual ens podria demostrar que és una paraula que discretitza entre les dues classes. En canvi, si mirem el contrari, *bad*, veiem que les freqüències són bastant similars, així que pot ser que aquest adjectiu no sigui tan determinant.

Tot i això, el fet que la llista d'adjectius sigui tan similar ens pot fer pensar que és possible que les prediccions s'estiguin veient empitjorades per culpa del soroll de paraules que no aporten tanta informació sobre la polarització, pel fet que, si no, el normal potser hauria sigut trobar més adjectius negatius a les prediccions negatives i adjectius positius a les positives.

5. Models no supervisats

5.1. Utilització de Lesk

Tot i que es proposa utilitzar UKB per obtenir els *synsets* de les paraules, es va recórrer a la utilització de *Lesk* per fer aquesta mateixa tasca a causa de les dificultats tècniques i limitacions que aporta el primer. Tenint en compte això es va realitzar aquesta segona part de la pràctica el més fidel possible.

5.2. Enfocament

La idea inicial pel desenvolupament d'aquest estudi va recaure en extreure *synsets* per cada paraula de cada frase de manera desambiguada i després obtenir els valors positius i negatius. En acabar l'anàlisi sencera de la frase compara la suma de valors positius i negatius, definint la polaritat de la *review*. En una primera instància esperarem pitjors resultats que en la primera part a causa de l'aparent falta d'intel·ligència del model.

5.3. Codi i procés

En aquesta part tenim dos codis principals, el del mateix model i el de l'anàlisi dels resultats. En aquesta part de la pràctica ens centrarem més en el conjunt de test tot i que també es realitzarà l'estudi al *train*. Al principi trobem la part de lectura de dades i preprocessament que ja ha estat explicada. Seguidament, veiem un conjunt de funcions definides per la posterior execució del model. Tenim 2 funcions principals i 3 auxiliars. Les dues primeres és on recau la major part del pes del procés.

La primera (*analyze_sentence_with_scores*) analitza la frase donades unes certes categories. Això ho fa mitjançant la crida a la segona funció principal (get_sentiwordnet_score) que analitza donada la frase i el *post tag* cada una de les paraules de la frase retornant el seu valor positiu i el negatiu amb l'ajuda del *senti_synset* de *Senti_wordnet*. Per tant, la primera

funció va acumulant aquests resultats, que ara entrarem més en detall de com s'obtenen, i en acabar la frase, com hem explicat abans, decideix si és positiva o negativa amb una simple comparació, cosa que no és molt intel·ligent. Si la suma de positius és més gran que la de negatiu la *review* és positiva i viceversa.

En l'anàlisi de la paraula en vam trobar que hi havia vegades que mitjançant *Lesk* no es trobava synset o que no es trobava *senti_synset*. En una primera aproximació es va decidir que quan això passes la funció retornés valors nuls d'ambdues polaritats, per tal d'afectar el menys possible a l'anàlisi. No obstant es va explorar aplicar altres estratègies per tal de trobar una millor solució i a més dur a terme una anàlisi més fidedigna.

Una d'aquestes millores va ser la cerca dels hiperònims, hipònims i lemes més semblants a la paraula i el seu *pos tag* quan inicialment no es trobés synset amb *Lesk*. A més com últim recurs es va buscar una alternativa pel cas que de cap de les maneres es trobés *synset* mitjançant *Vader* i el seu *polarity scores*. Agafant únicament els valors positius i negatius de l'anàlisi *Vader* fèiem més complet l'anàlisi. Indagant més vam trobar que potser era poc informatiu, ja que al només passar-li la paraula i perdre context potser la polaritat no era la més correcta. No obstant això, vam pensar que encara era una bona mesura per aprofundir el nostre estudi i els resultats ho demostren com més endavant ensenyarem. A més tenim una funció que ens permet compara diferents maneres d'etiquetes pel *postagging*, una formalitat si més no.

Finalment, tenim la funció pròpia del model que aglutina totes les anteriors. Bàsicament, és una funció que donada uns corpus (*train,test*) unes categories (adjectius, noms, verbs i adverbis) i una combinació d'aquestes du a terme l'anàlisi comentada abans *review* per *review* i a més dona l'*accuracy*, la matriu de confusió, juntament amb el número de *missclassifications*.

Ara passem a la part de l'anàlisi d'errors. Tenim un petit *script*, al mateix fitxer que l'estudi anterior, on es printeja les *reviews* que han estat mal classificades per la millor combinació de categories. A més, el seu valor positiu i negatiu. En aquest cas, com hem comentat abans, ens centrarem en el test per poder comparar-ho amb el de la part supervisada.

Amb un altre *script* analitzem el fitxer de text que generem abans i obtenim per les dues classes (True labels), la mitjana de diferència entre valors, la mediana i la moda. Més endavant ho analitzarem amb més profunditat.

5.4. Resultats

Com hem comentat abans, es va realitzar el procés considerant com a valors nuls els *synsets* no trobats. Es va executar el codi per diferents combinacions de categories i els resultats van ser aquests:

```
Running analysis for: All Categories

Training Accuracy: 0.555625
Training Confusion Matrix:
[[229 571]
 [140 660]]

Testing Accuracy: 0.5725
Testing Confusion Matrix:
[[ 60 140]
 [ 31 169]]

Number of misclassified training sentences: 711
Number of misclassified testing sentences: 171
--------------------------------------------------------------------------------------------------

Running analysis for: Without Verbs

Training Accuracy: 0.594375
Training Confusion Matrix:
[[512 288]
 [361 439]]

Testing Accuracy: 0.635
Testing Confusion Matrix:
[[137  63]
 [ 83 117]]

Number of misclassified training sentences: 649
Number of misclassified testing sentences: 146
--------------------------------------------------------------------------------------------------
```

Figura 12: Totes les categòries i sense verbs

```
Running analysis for: Without Adverbs

Training Accuracy: 0.55875
Training Confusion Matrix:
[[182 618]
 [ 88 712]]

Testing Accuracy: 0.5525
Testing Confusion Matrix:
[[ 41 159]
 [ 20 180]]

Number of misclassified training sentences: 706
Number of misclassified testing sentences: 179
--------------------------------------------------------------------------------------------------------

Running analysis for: Without Adverbs and Verbs

Training Accuracy: 0.589375
Training Confusion Matrix:
[[435 365]
 [292 508]]

Testing Accuracy: 0.5975
Testing Confusion Matrix:
[[113  87]
 [ 74 126]]

Number of misclassified training sentences: 657
Number of misclassified testing sentences: 161
--------------------------------------------------------------------------------------------------------
```

Figura 13: Sense adverbis i sense verbs i adverbis

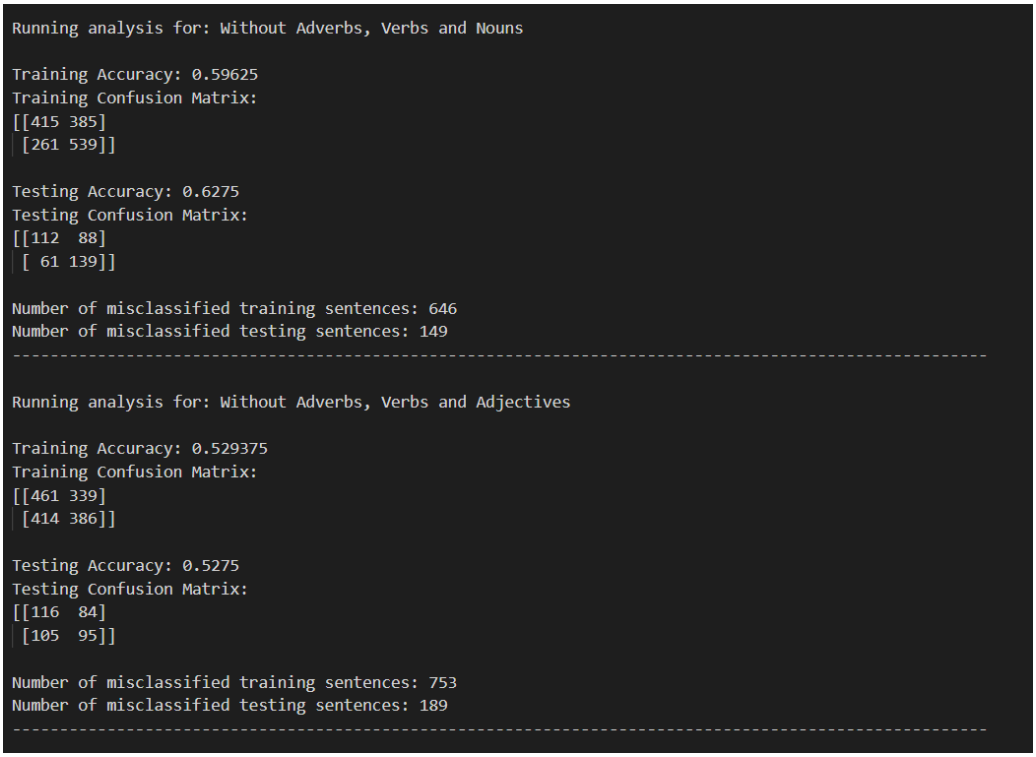


Figura 14: Només adjectius i només noms

Com es pot observar en totes les combinacions els resultats són relativament semblants. No obstant es pot observar que els millor resultats són obviant els verbs amb una *accuracy* de 0.635 al test, a més és on veiem la millor matriu de confusió. A la resta de combinacions veiem pitjors *accuracies* i considerablement pitjors matrius de confusió. Per exemple quan utilitzem totes les categories es prediuen molt millor les negatives, però les positives s'equivoca en la gran majoria. Passa el mateix sense els adverbis. Sense adverbis i verbs també tenim bons resultats però no tant com el sense verbs, potser per la falta d'informació. Per últim, provem amb només noms i només adjectius on veiem que només noms els resultats són considerablement pitjors que amb només adjectius. Probablement perquè els adjectius de manera única expressen més polaritat que només els noms.

Com hem comentat abans vam explorar altres estratègies, buscant alternatives als synsets no trobats. Aquestes van donar els seus fruits, tot i que la millora no va ser considerable. Aquests van ser els resultats:

```
Anàlisis de: All Categories

Training Accuracy: 0.5825
Training Confusion Matrix:
[[218 582]
 [ 86 714]]

Testing Accuracy: 0.6025
Testing Confusion Matrix:
[[ 56 144]
 [ 15 185]]

Number of misclassified training sentences: 668
Number of misclassified testing sentences: 159
-----------------------------------------------------------------------------------------------------

Anàlisis de: Without Verbs

Training Accuracy: 0.631875
Training Confusion Matrix:
[[453 347]
 [242 558]]

Testing Accuracy: 0.6675
Testing Confusion Matrix:
[[124  76]
 [ 57 143]]

Number of misclassified training sentences: 589
Number of misclassified testing sentences: 133
-----------------------------------------------------------------------------------------------------
```

Figura 15: Totes les categòries i sense verbs

```
Anàlisis de: Without Adverbs

Training Accuracy: 0.56875
Training Confusion Matrix:
[[172 628]
 [ 62 738]]

Testing Accuracy: 0.58
Testing Confusion Matrix:
[[ 45 155]
 [ 13 187]]

Number of misclassified training sentences: 690
Number of misclassified testing sentences: 168
-----------------------------------------------------------------------------------------------------

Anàlisis de: Without Adverbs and Verbs

Training Accuracy: 0.623125
Training Confusion Matrix:
[[411 389]
 [214 586]]

Testing Accuracy: 0.6375
Testing Confusion Matrix:
[[101  99]
 [ 46 154]]

Number of misclassified training sentences: 603
Number of misclassified testing sentences: 145
-----------------------------------------------------------------------------------------------------
```

Figura 16: Sense adverbis i sense verbs i adverbis

```
Anàlisis de: Without Adverbs, Verbs and Nouns

Training Accuracy: 0.626875
Training Confusion Matrix:
[[392 408]
 [189 611]]

Testing Accuracy: 0.655
Testing Confusion Matrix:
[[100 100]
 [ 38 162]]

Number of misclassified training sentences: 597
Number of misclassified testing sentences: 138
-----------------------------------------------------------------------------------------------------

Anàlisis de: Without Adverbs, Verbs and Adjectives

Training Accuracy: 0.546875
Training Confusion Matrix:
[[454 346]
 [379 421]]

Testing Accuracy: 0.525
Testing Confusion Matrix:
[[111  89]
 [101  99]]

Number of misclassified training sentences: 725
Number of misclassified testing sentences: 190
-----------------------------------------------------------------------------------------------------
```

Figura 17: Només adjectius i només noms

Com podem observar la millor combinació de categories continua sent la que no incloem els verbs en l'anàlisi. Com hem comentat, s'ha vist una millora tot i que no és substancial. Passem de classificar de manera incorrecta 146 *reviews* a 133, augmentant l'*accuracy* per 0.04. La resta de combinacions també milloren en general, però es mantenen en quasi el mateix ordre que l'anterior. Per tant confirmant de nou que quan s'analitzen noms, adjectius i adverbis ens trobem amb el millor estudi.

5.5. Anàlisis d'errors

Per la següent part, l'anàlisi d'errors, en centrarem en el conjunt de test de la millor combinació que hem decidit a l'apartat anterior. Sabent com s'ha decidit la polaritat d'una *review* hem volgut analitzar a fons com havia quedat aquesta comparació en les frases que s'han classificat malament. Calculant la distància mitjana entre l'acumulació positiva i la negativa. A més calculem la mediana i la moda entera. Aquest son els resultats:

```
True Label = 1, Error de classificació:
  - Distància mitjana: 3.5073859649122805
  - Distància mediana: 2.75
  - Distància moda (int): 0

True Label = 0, Error de classificació:
  - Distància mitjana: 3.6847105263157895
  - Distància mediana: 3.0625
  - Distància moda (int): 1
```

Figura 18: Resultats per les mal classificades en el test

Podem observar que per ambdues classes les mitjanes són semblants, tot i que cal remarcar que quan la *review* originalment és positiva la mitjana de distància és més petita, igual que la mediana. També és bastant informatiu saber que la moda entera són 0 pels positius i 1 pels negatius, denotant que en la majoria dels casos aquesta comparació estava molt igualada. Cosa que es pot donar quan el text té certa ambigüitat en vers la polaritat.

Després d'això, vam trobar interessant comparar aquestes distàncies amb les de les opinions correctament classificades, per tal de veure si hi ha una gran diferència, així que vam fer el mateix exercici amb les prediccions correctes. Tal com podem veure a la figura 19, els resultats van ser els esperats, i les distàncies a les prediccions correctes són molt més grans.

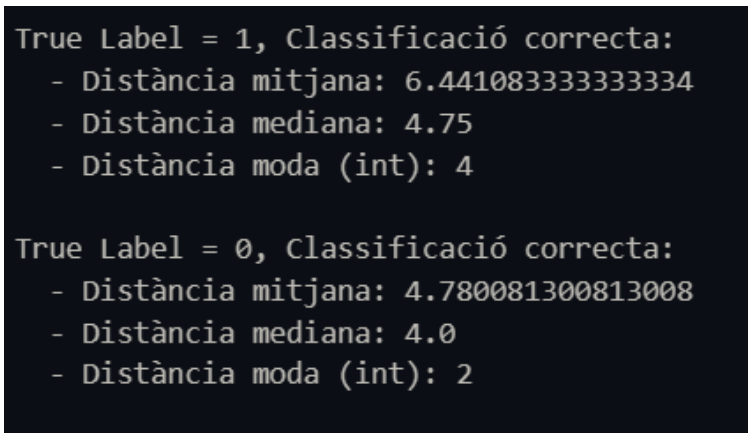


Figura 19: Resultats per les correctament classificades en el test

6. Possibles millores

Al llarg d'aquest treball hem tingut molts moments en què hem tingut dubtes sobre com abordaríem diferents problemes o com implementaríem certs conceptes. És per això que hi ha un gran nombre d'idees que han quedat pel camí o que han aparegut més tard i que no s'han pogut materialitzar per diferents motius, ja sigui per temps, per recursos, etc.

Un dels principals, i que ja hem comentat en aquest informe, hagués sigut una millor cerca d'hiperparàmetres en el model supervisat. En aquest cas, hem hagut d'abordar-ho d'una manera més general, però amb uns millors recursos hauria sigut interessant tunejar els models de predicció i el de vectorització a la vegada, provant moltes combinacions d'hiperparàmetres, per tal de donar amb el millor model.

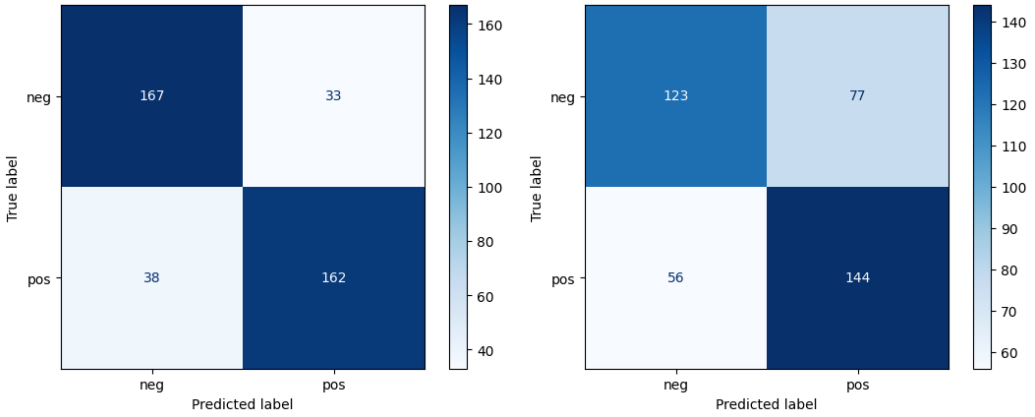
Per altra banda, pel que fa als models no supervisats, tot i que la nostra decisió per provar diferents models va ser provar d'excloure diferents categories, també hagués sigut interessant provar un sistema de pesos, donant més o menys importància a certes categories. Aquesta és una idea que se'ns va ocórrer més tard i que ens hagués agradat provar.

Finalment, també vam pensar que, com cada model té errors diferents, potser seria possible combinar les dues prediccions per tal d'obtenir un millor resultat. Per fer-ho, vam pensar que, per a cada opinió, ens podríem quedar amb la predicció del no supervisat si la diferència entre el score positiu i el negatiu és molt gran, el qual vol dir que té més seguretat que la predicció és bona, i amb la del supervisat si la diferència és petita. En el nostre cas, la implementació proposada no obtenia millors resultats, però, de cara a una ampliació d'aquest treball en un futur, és una idea en la qual podríem aprofundir més.

7. Comparació entre model supervisat i no-supervisat

El primer que podem veure, per tal de comparar els models, és les accuracies i les matrius de confusió d'aquests dos. Recordem que el millor model supervisat va ser la regressió logística, la qual va obtenir una accuracy del 82 % al test, i el millor model no supervisat va ser el que no inclou els verbs, que va obtenir una accuracy del 67 % amb aquest mateix conjunt. Per tant, podríem determinar que el rendiment dels models supervisats ha sigut millor que el dels no supervisats.

Aquesta diferència la podem apreciar, també, observant les matrius de confusió, on s'aprecia clarament com el nombre d'errors del mètode no supervisat és molt major. D'aquest també és interessant el fet que les opinions negatives es van predir notablement pitjor, amb 20 errors més respecte a les positives.



(a) Matriu del model supervisat **(b)** Matriu del model no supervisat

Per acabar, també vam trobar que seria interessant veure si els dos models es van equivocar a les mateixes opinions o no. Per això, vam decidir fer un petit codi per comparar els errors. Els resultats obtinguts els podem veure a la taula 3 i podem observar com 30 errors van ser compartits entre els dos models. Això vol dir que el model supervisat comparteix menys de la meitat dels seus errors amb el no supervisat, el qual ens pot fer veure que tenen patrons d'error diferents i interpreten les opinions de manera diferent.

Errors comuns	Errors només al supervisat	Errors només al no supervisat
30	41	103

Tabla 3: Longitud de les prediccions correctes i incorrectes

8. Conclusions

Un cop acabat aquest treball i analitzats tots els resultats obtinguts, hem pogut extreure diverses conclusions sobre l'eficàcia i l'eficiència d'aquests mètodes en la detecció d'opinions positives i negatives.

En primer lloc, hem observat que el preprocessament de les dades és una fase clau per a l'èxit de qualsevol model. És per això que hem aplicat diferents transformacions a les opinions abans d'utilitzar-les, com ara la lematització o l'eliminació de les *stop words* per treure el soroll.

Pel que fa als models de *Machine Learning*, hem comprovat que la regressió logística ha estat el millor model en termes de extitaccuracy amb un 83.9 %, seguida de Random Forest amb un 82.6 % i Naïve Bayes amb un 81.7 %. Els millors resultats s'han obtingut amb la vectorització de TF-IDF, el qual indica que la ponderació de paraules en funció de la seva importància en els documents ha millorat el rendiment dels models.

D'altra banda, també s'han provat models no supervisats, els quals utilitzen llibreries per a calcular, per a cada paraula, una puntuació que mostra com de positiva o negativa és. En aquest cas, l'accuracy més alta obtinguda ha sigut del 66 %, el qual ens mostra que el fet d'entrenar els models amb les etiquetes permet obtenir millors resultats.

Finalment, cal destacar la importància de la prova de diferents models i la selecció d'hiperparàmetres, ja que ajustant-los adequadament s'ha aconseguit millorar la precisió dels models. Aquests resultats ens permeten concloure que la detecció d'opinions mitjançant mètodes supervisats és una eina eficient i amb potencial per ser aplicada en escenaris reals com l'anàlisi d'opinions de productes o negocis.