Amazon Product Reviews Sentiment Analysis

Guillem Sallas, Arnau Busquets, David Martí, Jordi Auñon
Aprenentatge Computacional 2023-2024

Índex

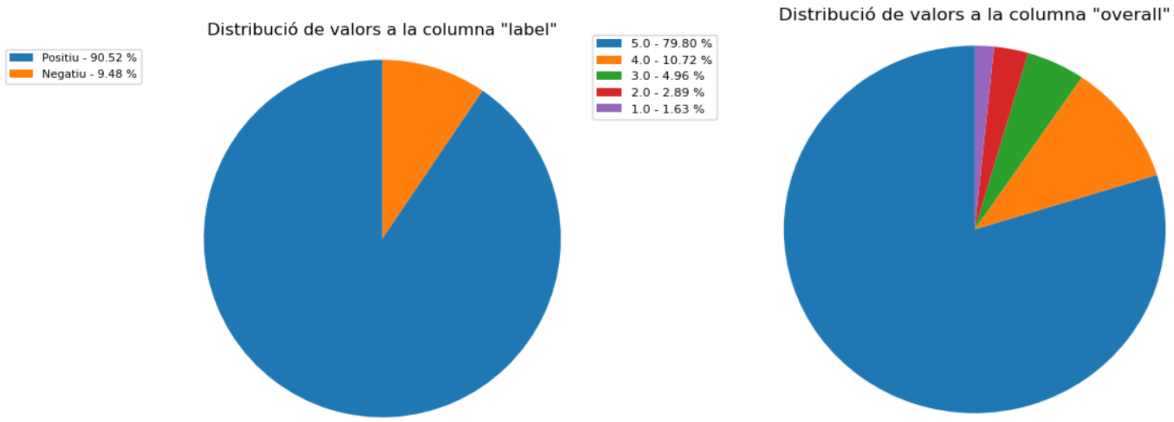
Les dades

El nostre dataset es tracta d'un dataset que contempla diferents paràmetres que aporten informació sobre "reviews" de productes venuts a amazon. Respecte als paràmetres inclosos als datasets tenim de molts tipus com el temps des de que es va fer la review fins el dia de la consulta de les dades per crear el dataset fins la puntuació "overall" (1-5) que el client va posar al producte.

Pel nostre projecte i basant-nos en els models que volem estudiar i per recomanació del professor vam decidir acotar els paràmetres emprats al text de la review "reviewText" i la puntuació "overall" donada pel client.

Com ha conclusió després d'haver provat diferents formes de classificar les reviews vam decidir classificarles amb dos diferents "labels" "pos", "neg" positives les reviews amb overall 4,5 i negatives amb 1,2,3.

Per veure quantes reviews amb cadascuna de les puntuacions i amb les dos labels ho vam mostrar amb els següents plots:



Analitzant els plots podem veure que tenim un gran desbalanceig al dataset amb les puntuacions que respectivament afecta a les labels. Tenim un 79,8% de valoracions amb puntuació = 5 i 10.72% de valoracions = 4 generant que el dataset tingui un 90.52% de reviews amb label "pos".

Text Processing

Per tractar les dades i poder aplicar els nostres models a les reviews hem d'aconseguir un text fàcilment tractable. En aquesta part entra el "text processing" un procés en el que preparem el text per tal de poder aplicar el models a aquest text. Per generar aquest text que es tractarà hem emprat els següents mètodes.

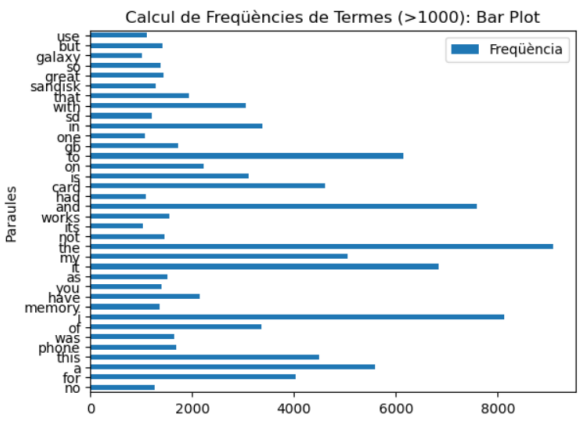
- Normalització del text:
  En aquesta part del processament s'eliminen les lletres en majúscula i es substitueixen per la lletra corresponent en minúscula.

- Eliminació de nombres i signes de puntuació:
  Consisteix en l'eliminació dels nombres i els signes de puntuació per tal de deixar el text el més net i concís possible.

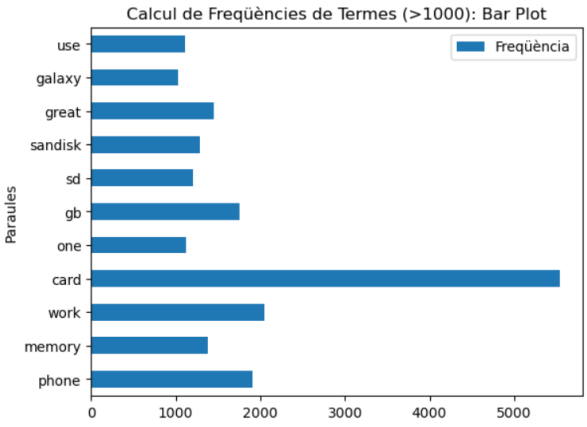
- Eliminació de les "Stopwords" i paraules estranyes:
  Les "stopwords" son paraules com preposicions i articles que es repeteixen molt als textos però que no aporten significat per tant a l'hora d'aplicar els models es millor retirar-les perquè nomes generaran soroll al nostre estudi. Per altra banda, les paraules considerades com a rare words son paraules que es repeteixen molt poc que no hi ha benefici a prendre-les en compte.

- Lematització:
  L'aplicació d'un lematitzat a les paraules és útil per treballar a partir de la paraula arrel d'una paraula i no tenir diferents paraules que es tracten com diferents però son la mateixa conjugada de diferents formes.

Gràfiques on es pot veure el canvi en el preprocessament i el postprocessament:



Preprocessament



Postprocessament

Vectorització

Diferents mètodes de vectorització

Un cop processat el text cal vectoritzar-lo per així poder tindre'l representat de manera numèrica per així poder entendre l'algorisme les dades. El pas de vectoritzar consisteix en transformar aquest text seguint un mètode i obtenir com a resultat un vector de nombres.

Els 3 mètodes que hem escollit per veure quin ens dona millors resultats al nostre dataset son:

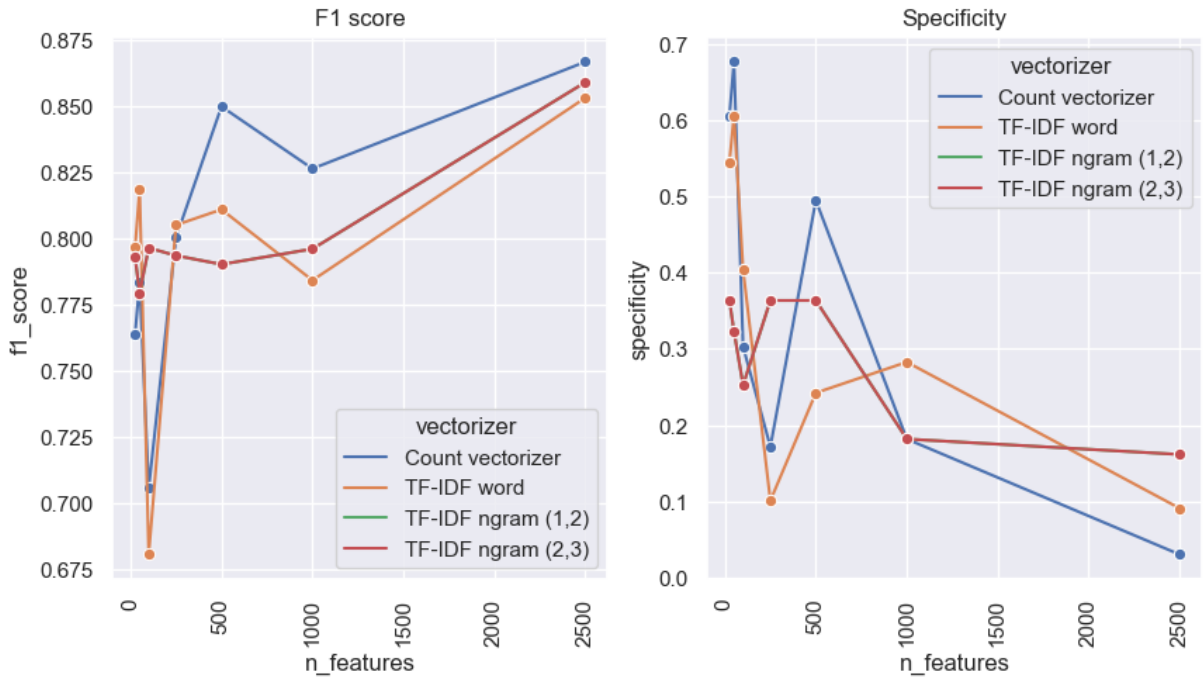
- Count Vectors: Cada vector és un document i cada punt la freqüència d'una paraula en aquell document.
- TF-IDF word: Cada valor del vector es calcula mitjançant la multiplicació de dos valors: la freqüència de la paraula (TF) i la freqüència inversa del document (IDF).
- TF-IDF n-gram: Es calcula de la mateixa forma que "word" però es refereix a un conjunt de paraules consecutives com a unitat.

Comparació de resultats

Per poder escollir quin mètode s'adapta més al nostre problema hem treballat sobre el mòdel de logistic regression amb les dades balancejades (s'explica en el següent apartat) i hem tret conclusions apartir de com fa les prediccions tant als negatius com els positius.

Respecte cada mètode de vectoriztació també els hem provat amb un rang de valors en certs paràmetrs per així millorar el seu rendiment.

Comencem pel paràmetre n_features que consisteix en dir el tamany màxim que ha de tenir cada vector en la vectorització, és a dir, el tamany de les files.
Per poder veure l'efecte d'aquest paràmetre hem fet aquestes gràfiques on veiem la variació de l'F1-Score i la specificity:



En aquesta gràfica podem veure el valor de la mètrica en en el rang de [25, 50, 100, 250, 500, 1000, 2500] que son valors que per Count Vectors i TF-IDF word son adients perquè pot arribar com a màxim a 2620, però en canvi per n-gram pot semblar que sigui dolents però com hem pogut analitzar amb els resultats quants més n-grames hi ha pitjors resultats dona.

Respecte els resultats de les gràfiques podem veure com n-grames (de fet son tant semblant que es solapen i no es poden distingir) ens donen els pitjors resultats en totes dues, probablament degut a l'informació que es perd al fer les agrupacions, ja que genera molt de soroll i pot ser que doni suposicions errònies. A més, les agrupacions de paraules poden no capturar adequadament les relacions semàntiques complexes entre les paraules

Després podem tenim a TF-IDF word i Count Vectorizer que son semblants ja que tots dos ens donen una specificity molt alta quan el nombre de features es baix i un F1-Score baix. Això es degut a que prediuen millor els negatius per F1-Score ja que no tendeix a posar sempre positiu, el qual pot ser útil, a l'hora de fer l'analisis ja que ens confirma que el balanceig serveix a l'hora de predir.

Count vectorizer	2500	0.866637	0.030303	TF-IDF word	2500	0.853063	0.090909

D'entre aquests mètodes ens quedem amb Count Vectorizer amb un nombre de features de 25 perquè és la que ens dona millors prediccions i encara que amb el mateix paràmetre tenim més F1-Score amb TF-IDF Word valorem més el balanceig de les dades.

Ja escollit aquest mètode, ara hem de trobar altres paràmetres per les prediccions.

Una opció que pensabem que seria útil era min_df i max_df que es refereix a la freqüència respecte els documents que han de tenir com a mínim i màxim els features per ser afeguits dins del vector. Però, el resultat ha sigut el següent:

min_df min_df

No hem trobat cap rang de valors on canvi la gràfica sense donar error (perquè no hi ha features que compleixin l'interval) ja que al tenir com a màxim 25 features per asegurar un bon balanceig, la freqüència d'aquells és molt similar i marcar més o menys l'interval no canvia en res a les prediccions.

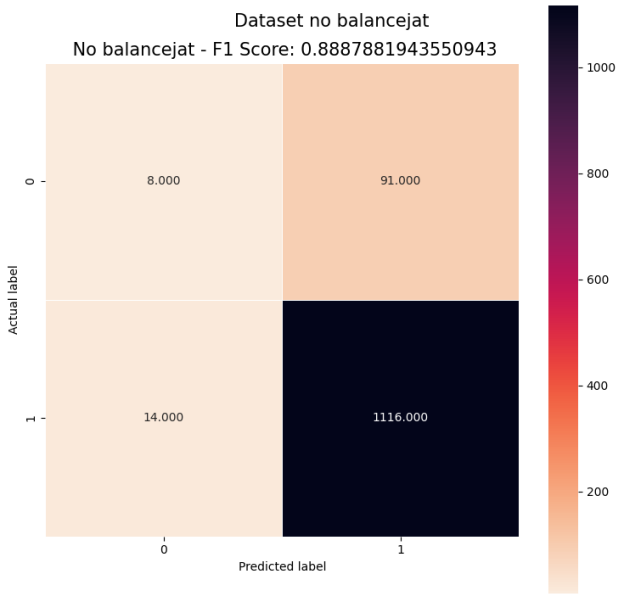
Per tant la conclusió és que el millor mètode de vectorització pel nostre dataset és Count Vectors amb màxim de features d 25 per aconseguir resultats al màxim de balancejats i a la vegada força bons.

.

Balanç del dataset

Com hem vist, aquest és un dataset desbalancejat. És per això que, fixant un tipus de model logarítmic i una vectorització de tipus Count Vectors, veurem com canvien els resultats de la matriu de confusió i F1 Score a l'intentar balancejar les dades.

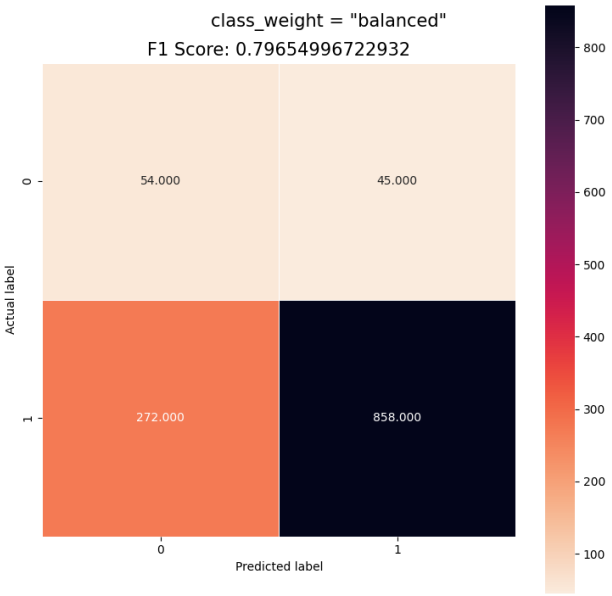
Usant un model logístic sense balancejar obtenim el següent:



Podem veure molt bons resultats entre reviews positives però molt dolents entre negatives (només prediu 8 bé de 99). Això produeix un f1 score molt gran ja que la gran majoria de comentaris positius son ben classificats però una capacitat de diferenciar negatius gairebé nul·la.

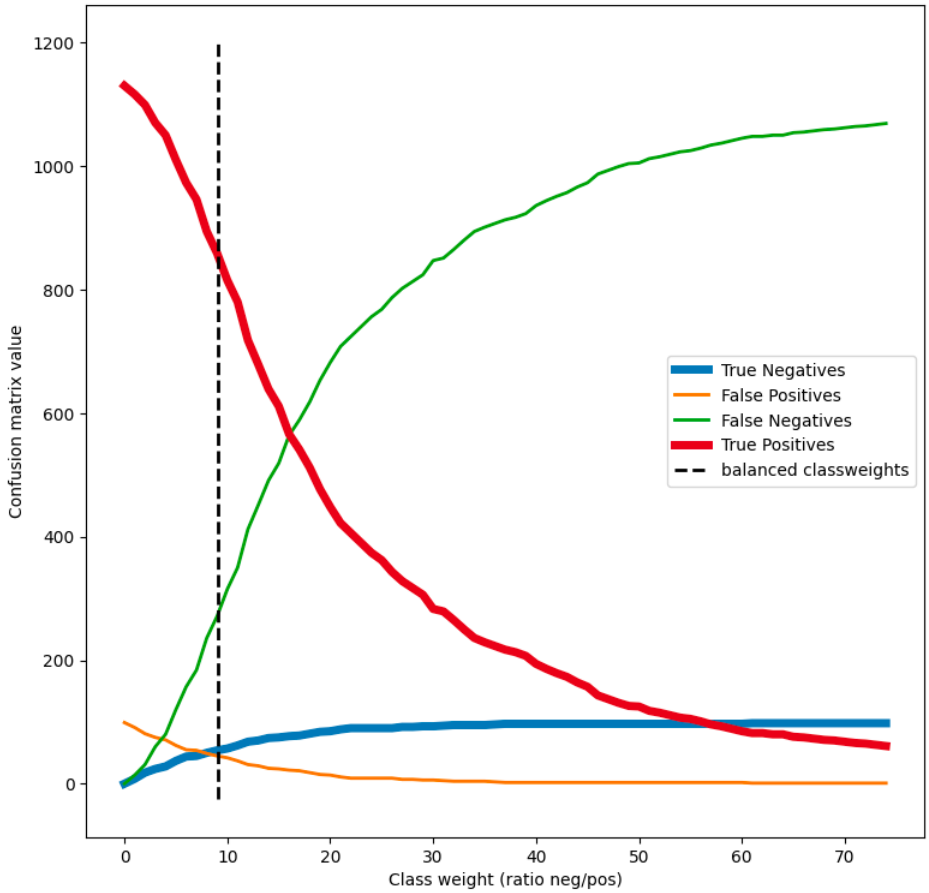
Ús del paràmetre class_weights

Aplicant aquest paràmetre deixem que la funció calculi uns pesos adequats per als conjunts positius i negatius en l'entrenament del model per tal de que les prediccions no siguin afectades per el balanceig del data set de l'entrenament.



Podem veure molt clarament que els resultats entre les reviews negatives són molt més bons, ara ja es supera el 50 per cent d'encerts mentre abans no arribava al 10. Tot i així, vegem que F1 Score és més baix i això és perquè hem perdut moltes de les prediccions TP, reviews positives que ara es classifiquen com a negatives (més que no pas reviews negatives que hem aconseguit classificar correctament. Tot i així cal que quedi clar que aquest és un model molt més capaç davant qualsevol tipus de comentari, i ja no tendeix automàticament a classificar-lo com a positiu.

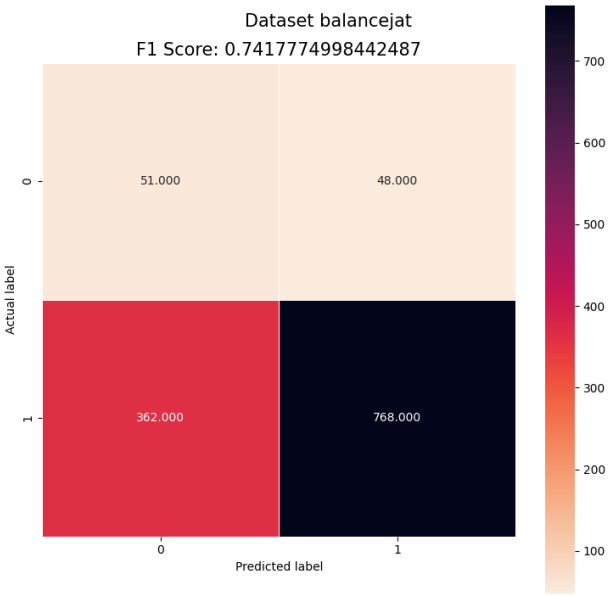
Per a veure clarament la capacitat del paràmetre class_weights, es mostra a continuació com varien els valors de la matriu de confusió segons la ratio de class_weights (en aquest cas, el nombre a l'eix x correspon a quantes vegades més pes adquireixen les instàncies negatives respecte les positives). En negre, queda marcat el punt que s'aplica quan el paràmetre class_weights és 'balanced'.



L'objectiu és aconseguir TP i TN, marcats amb una línia més gruixuda en el gràfic. Si bé el punt 0 seria el que més resultats positius aconseguiria, serien tots sobre reviews positives, creiem que la línia negra marca un bon ajustament de pesos ja que aconsegueix fer créixer molt la specifity i uns pesos més extrems farien baixar massa els TP.

Creació d'un dataset 50-50

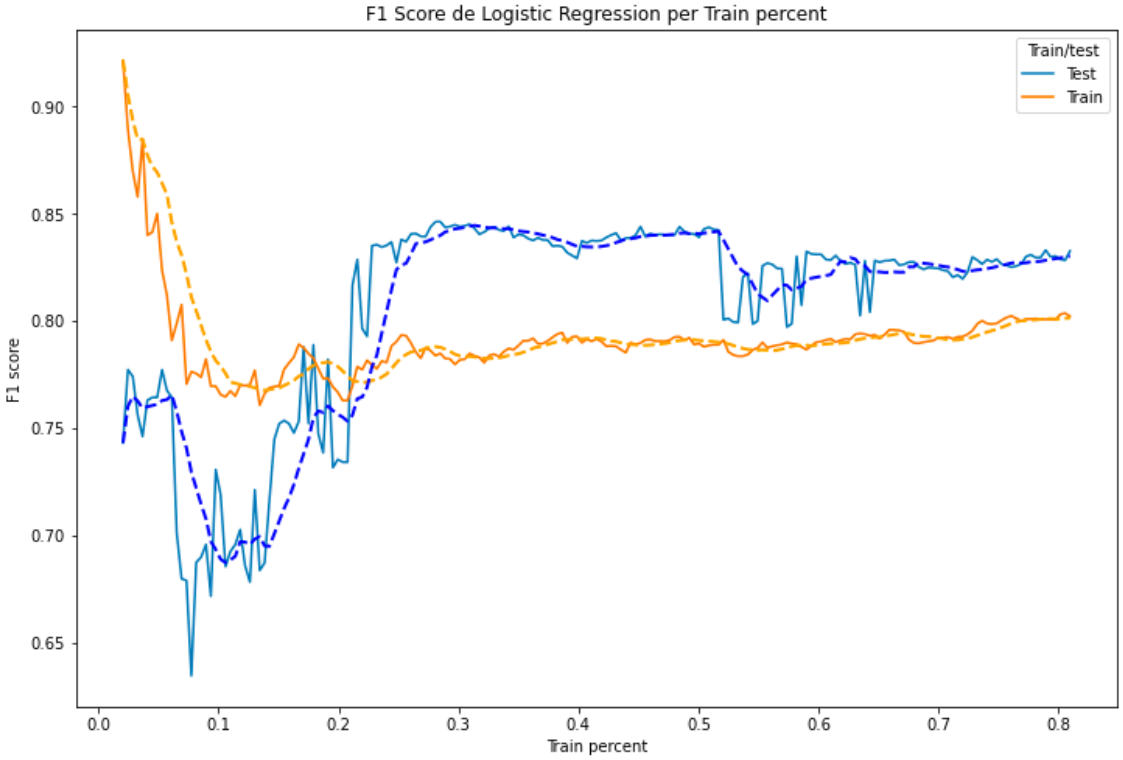
També hem provat la creació d'un dataset on la meitat de les dades són positives i l'altre negatives, hem entrenat amb aquest dataset el model i així en hem assegurat de que aquest no patiria cap efecte per desbalanceig. Aquesta ha estat la matriu de confusió i F1 Score resultants:

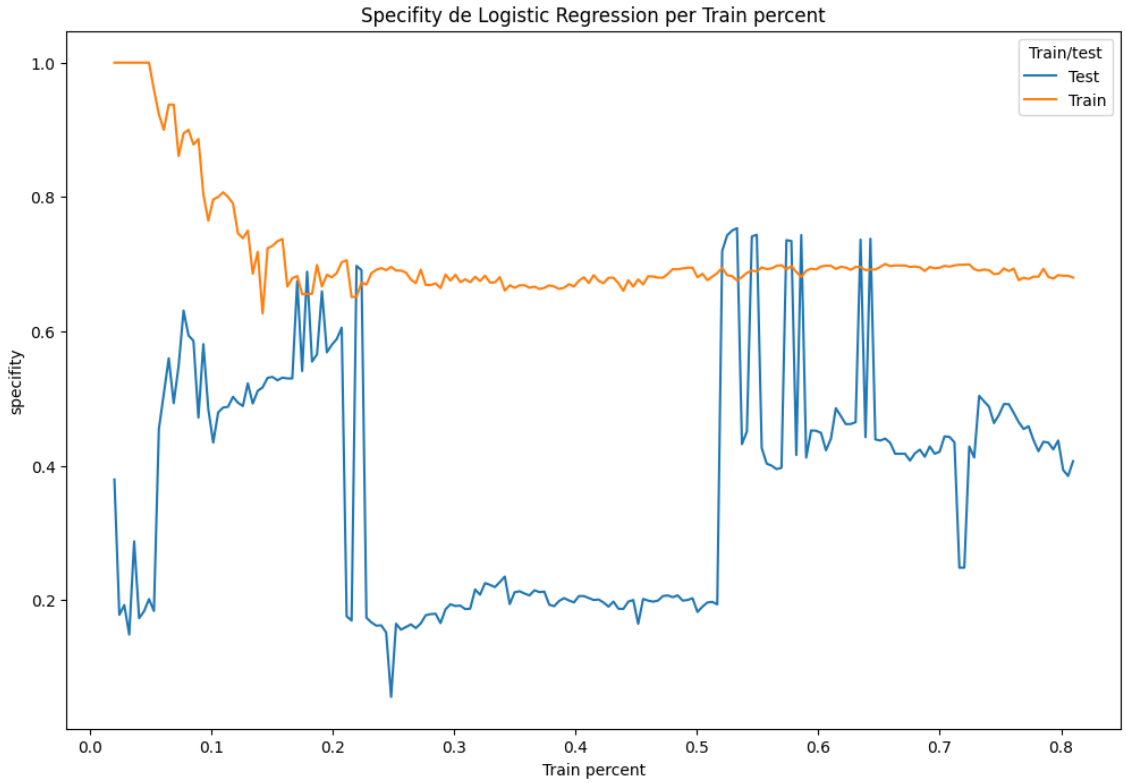


Podem apreciar uns resultats molt millors que en el dataset sense balancejar, pels mateixos motius explicats en el segon cas. Però no tant bons com en el dataset que hem balancejat usant la funció class_weights. Tot i que no és una gran diferència, entenem que si es balancegen els pesos, no tenim un excés de dades de reviews positives així que **seguirem al projecte aplicant la normalització class_weights = 'balanced'.**

Efecte de la quantitat de dades en la qualitat de la predicció

L'objectiu següent és veure com evolucionen les mètriques F1 score i Sensitivity a mesura que augmentem el volum de dades d'entrenament. Ho farem predient sobre les dades test i sobre les dades train. Cal recordar que estem entrenant un dataset amb un 90% de files amb label positiu i només un 10% negatiu.





Podem veure que sobre les dades train, i poca quantitat de dades, ambdues mètriques són molt bones, ja que és molt fàcil predir. En canvi, quan les dades d'entrenament creixen, les dues sofreixen una caiguda fins a estabilitzar-se, en els dos casos al voltant del 10%.
Òbviament aquestes dades són irreals per a una predicció real, ja que les dades sobre les que avaluem són les mateixes amb les que hem entrenat el model. Cal remarcar que degut a la petitesa del nostre dataset, els resultats són molt volàtils, i hem de fixar-nos en una tendència general ja que pujades i baixades poden ser causades simplement per casualitats i semblances entre reviews.

Comparem els resultats amb els de les dades test, d'aquesta manera podríem veure si en algun moment ajustem el model massa a les dades train i no a les de test. Per a l'F1 Score, observem una tendència clarament positiva, també es pot veure un resultat major al de les dades train, però no és de clau importància ja que no és del tot significatiu, i no passa en totes les mètriques. Per a la specifity, que ens marcà el percentatge de reviews negatives que han estat ben classificades, veiem que la gràfica es desconcertant. Realment cal dir que amb un dataset de mida major, la specifity milloraria, però en aquest cas, les dades negatives son només el 10% del dataset de menys de 4000 línies, un total de 366 reviews negatives, si d'aquestes entrenem el model amb una part només, i balancegem per a que aquestes tinguin un pes major, podem

dir que és normal que es creïn grans variacions ja que cada review té un valor molt gran dins del conjunt.

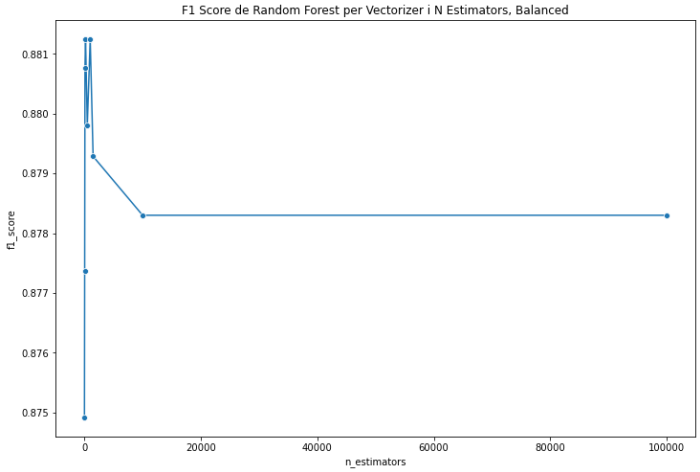
Efecte dels paràmetres al Random Forest

Nombre d'arbres

L'hiperparàmetre que especifica la quantitat de números d'arbres en un model Random Forest diem-li 'n_estimators'. Encara que pot semblar que millori el rendiment del model sempre que augmentéssim la quantitat d'arbres, això no és sempre cert, especialment quan es tracta de tasques específiques com en el nostre cas classificació de reviews d'Amazon.

Per exemple, hi ha un risc de generar sobreajustament, si s'augmenta la quantitat d'arbres , especialment si la base de dades en la que treballes el entrenament és petita.

També hi ha el cas de que hi ha una correlació entre el augment d'arbres pel model i el augment del temps en l'entrenament del model, el temps pot augmentar significativament i en casos pràctics això pot ser ineficient.



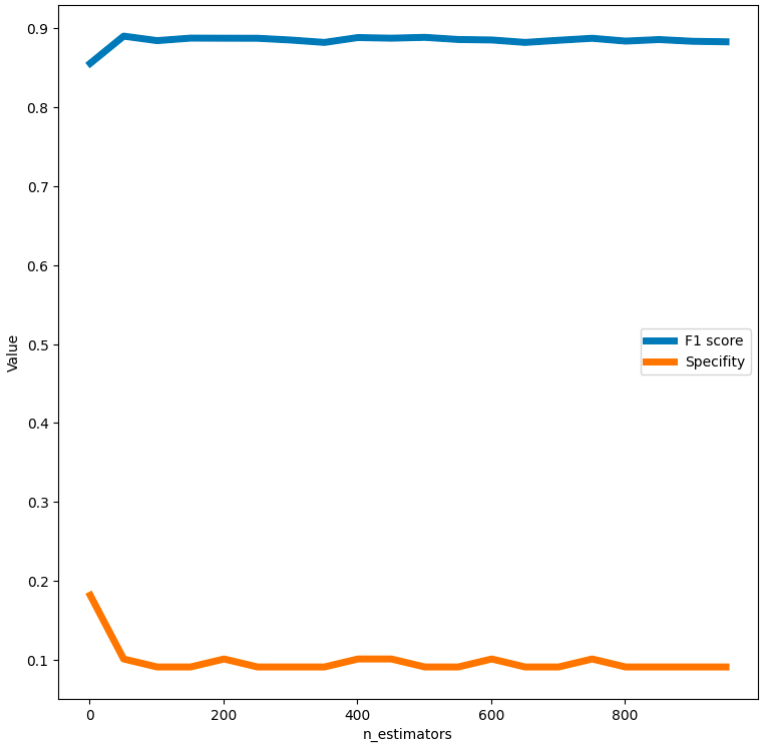
Valors del f1_score respecte diferents n_estimators:

	n_estimators	max_depth	f1_score
0	10.0	NaN	0.874914
1	50.0	NaN	0.877363
2	100.0	NaN	0.880770
3	200.0	NaN	0.881251
4	500.0	NaN	0.879802
5	1000.0	NaN	0.881251
6	1500.0	NaN	0.879287
7	10000.0	NaN	0.878301

```
8      100000.0        NaN  0.878301
```

Com veiem en aquesta gràfica l'hiperparàmetre del nombre d'arbres podria ser útil, però com hem estat comentant té inconvenients que fan que no sigui el que hem elegit.
Els valors del F1 score pel model Random Forest amb la mètrica Count Vectorizer i diferents nombres d'estimadors. A partir d'aquests resultats, podem observar que el rendiment del model sembla estabilitzar-se després d'un cert nombre d'estimadors
Amb només 10 estimadors, ja obteniu un rendiment relativament alt amb un F1 score de 0.874914. Això indica que fins i tot amb un nombre petit d'estimadors, el model és capaç de capturar bé les relacions en les dades.
A mesura que augmenta el nombre d'estimadors fins a 200, es produeix un augment progressiu en el F1 score, arribant al seu valor màxim de 0.881251. Aquest augment pot ser degut al fet que el model està millorant la seva capacitat de generalització amb més estimadors.
Seguidament, en arribar a 500 estimadors, hi ha una petita disminució en el F1 score fins a 0.879802. pràcticament insignificativa que podríem treure les conclusions de que pogués ser per un sobre ajustament o que el mètode es basa en una variabilitat aleatòria que fa que un cop arribat a un número d'arbres per paràmetre la precisió sigui casi constant a petites diferencies per la aleatorietat.
Finalment, sorprenentment, amb 1000 estimadors, el F1 score torna a pujar fins a 0.881251, igualant el valor màxim obtingut amb 200 estimadors. Això podria ser una indicació que, en aquest cas, afegir més estimadors no necessàriament millora el rendiment i fins i tot podria conduir a un lleuger sobre ajustament. Aquest repuntament de la precisió nosaltres concloem que pot ser donat per després d'un cert nombre d'estimadors, el model pot començar a sobre ajustar-se al conjunt d'entrenament.
Els 1000 estimadors podrien conduir a una certa reducció d'aquest sobre ajustament i, per tant, a una millora del rendiment en comparació amb 500. També, en alguns casos, afegir més estimadors pot introduir una complexitat addicional al model, i això pot afectar el rendiment. Pot ser que la combinació de factors en el teu conjunt de dades sigui més beneficiosa amb 1000 estimadors que amb 500, i que el model aconsegueixi millors resultats amb més estimadors.
La següent taula són els mateixos resultats però afegint els valors 1500, 10000, 100000. Això ens fa contextualitzar que el valor 1000 té més pes del que valoràvem per molt que pugui tenir tendència al sobre ajustament o a la fluctuació aleatòria. Però aquí veiem que al augmentar el número d'arbres es genera una pèrdua de rendiment fins a obtenir una constant, això pot ser

donat per un sobreajustament o el fet de que el ser un dataset de petites dimensions el augment dels arbres no genera millor rendiment.



S'ha utilitzat "max_features": 5000, "max_depth ": None

Com hem estat explicant els valors són molt semblants en el paràmetre n_estimators. És per això que quan fem una gràfica per el specifity veiem que per molt que 200 és el valor més òptim tots els valors són molt constants i per tant el paràmetre n_estimators realment no és de gran importància per determinar els millors valors per el f1_score i el specifity.

En resum, els resultats mostren una certa estabilització en el rendiment del model després d'un nombre suficient d'estimadors (aproximadament 200), amb fluctuacions petites que podrien ser atribuïdes a la variabilitat aleatòria o altres factors del conjunt de dades específic (aproximadament 1000). Per tant agafem el valor 200 com a òptim però podríem utilitzar qualsevol.

Taula gràfica de n_estimators, specifity i f1_score:

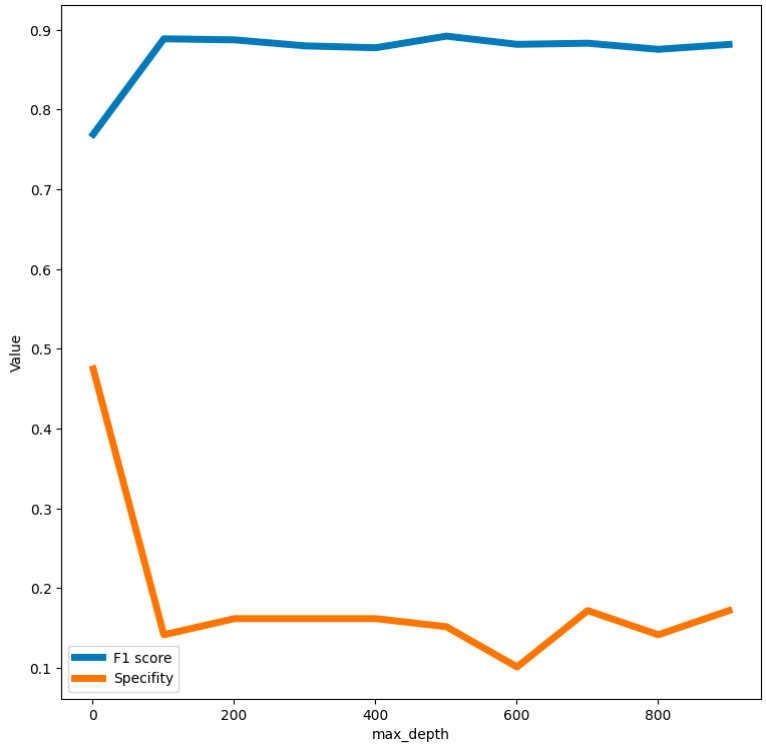
	n_estimators	max_features	Max depth	F1 score	Specifity
0	1	5000	None	0.855945	0.252525
1	101	5000	None	0.885471	0.101010
2	201	5000	None	0.881831	0.101010
3	301	5000	None	0.884167	0.090909

4	401	5000	None	0.883796	0.090909
5	501	5000	None	0.885032	0.090909
6	601	5000	None	0.883935	0.090909
7	701	5000	None	0.882916	0.090909
8	801	5000	None	0.884935	0.090909
9	901	5000	None	0.886556	0.090909

Profunditat Màxima

La profunditat controla la complexitat dels arbres del Random Forest. Amb una profunditat baixa, els arbres seran més somers i menys propensos a sobre ajustar-se als detalls específics del conjunt de dades d'entrenament. Això implica que el model és més generalitzable i menys sensible al soroll o a les característiques específiques del conjunt de dades d'entrenament. Augmentar la profunditat dels arbres pot conduir a overfitting, especialment si el conjunt de dades és relativament petit, com és el nostre cas.

També, els arbres poc profunds també són més eficients computacionalment, ja que requereixen menys temps per entrenar i predir.



S'ha utilitzat "max_features": 500, "n_estimators": 200.

(200 pq no hem tret conclusions de la primera gràfica

Seguint amb l'explicació, i amb el que sabem sobre el max_depth, veiem que a menor sigui aquest paràmetre millors resultats obtenim, ja que això genera un una millor adaptació, perquè com veiem al augmentar aquesta profunditat els valors es mantenen constants i molt dividits entre ells, generant això un sobre ajustament que no ens interessa tenir.

També veiem, que tenen una relació entre f1 score i specifity, que es veu molt visualment en la gràfica, que a mesura que puja el f1_score el specifity és cada cop més petit, i a la que un valor es manté constant el altre també segueix la mateix tònica.

Per tant, el valor més òptim per aquest paràmetre que nosaltres observem és de 5 en mas_depth, ja que veient la gràfica aconsegueix els millors resultats entre f1_score i el specifity, ja que tens uns bons valors entre els dos.

La taula dels valors de la gràfica de max_depth sobre f1_score i specifity:

	n_estimators	max_features	Max depth	F1 score	Specifity
0	10.0	5000.0	1.0	0.733179	0.474747
1	10.0	5000.0	101.0	0.886049	0.151515
2	10.0	5000.0	201.0	0.884213	0.151515
3	10.0	5000.0	301.0	0.879066	0.181818
4	10.0	5000.0	401.0	0.880446	0.151515
5	10.0	5000.0	501.0	0.882232	0.151515
6	10.0	5000.0	601.0	0.876833	0.131313
7	10.0	5000.0	701.0	0.884471	0.141414
8	10.0	5000.0	801.0	0.888724	0.141414
9	10.0	5000.0	901.0	0.890157	0.151515

Max_features

El hiperparàmetre max_features en el Random Forest determina el nombre màxim de característiques que el model considerarà per dividir un node durant la construcció de l'arbre. Aquest paràmetre és fonamental per diverses raons.

Principalment, limitar el nombre de característiques pot ajudar a controlar la variabilitat entre arbres. Això és important per prevenir sobre ajustament, ja que arbres amb moltes característiques podrien ajustar-se massa als detalls del conjunt de dades d'entrenament i no generalitzar bé a nous exemples.

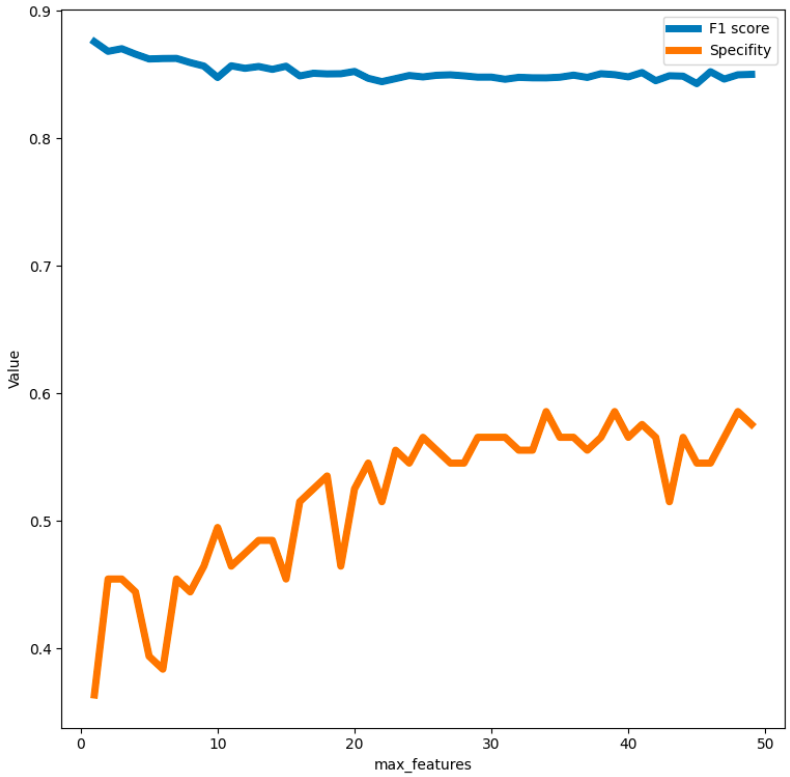
També, amb datasets grans amb moltes característiques (no és el nostre cas), el càlcul de les millors divisions per totes les característiques pot ser computacionalment costós. Reduir el nombre de característiques pot accelerar el procés d'entrenament sense sacrificar massa el rendiment.

Després, un Random Forest es beneficia de la diversitat dels arbres que el componen. Limitar les característiques pot assegurar que cada arbre es construeix amb diferents subconjunts de característiques, augmentant la diversitat global del conjunt de models.

Seguidament, també hi ha certs desavantatges que poden ser limitar el nombre de característiques pot fer que el model no sigui tant sensible als detalls subtils del conjunt de dades, cosa que pot ser desitjable o no segons el context.

Finalment, en conjunts petits de dades com el nostre, limitar el nombre de característiques podria conduir a la pèrdua d'informació important, ja que hi ha menys dades per aprendre les relacions entre característiques. Així que, en el nostre cas, és possible que el model no sigui tant robust ni generalitzat.

En resum, el paràmetre max_features és crucial per controlar la complexitat del model, millorar l'eficiència i promoure la diversitat en un Random Forest. No obstant, la seva influència depèn del context específic del conjunt de dades i pot requerir ajustaments experimentals per aconseguir el millor rendiment.



n_estimators max_features Max depth F1 score Specifity

0	200.0	1.0	5.0	0.885769	0.383838
1	200.0	2.0	5.0	0.872393	0.414141
2	200.0	3.0	5.0	0.864806	0.454545
3	200.0	4.0	5.0	0.863680	0.444444
4	200.0	5.0	5.0	0.867270	0.404040
5	200.0	6.0	5.0	0.859810	0.454545
6	200.0	7.0	5.0	0.853065	0.404040
7	200.0	8.0	5.0	0.854990	0.434343
8	200.0	9.0	5.0	0.858505	0.464646
9	200.0	10.0	5.0	0.861440	0.393939
24	**200.0**	**25.0**	**5.0**	**0.851774**	**0.565657**
48	200.0	49.0	5.0	0.846540	0.565657

Observant la gràfica amb els canvis de max_features i el seu impacte en el F1 score pel model Random Forest treiem algunes conclusions utilitzant la mètrica Count Vectorizer.

En general, es pot observar que a mesura que augmenta el valor max_features, el F1 score es manté constant i el specifity augmenta siginficativament. Això podria ser degut al fet que, amb més característiques disponibles per cada divisió de node, el model pot capturar millor les relacions i patrons en les dades d'entrenament, millorant així el seu rendiment en les dades de prova.

El valor òptim de max_features sembla ser al voltant de 25, ja que aquest valor està associat amb el màxim F1 score observat en 0.851774 de f1_score i de 0.565657 de specifity. Això suggereix que considerar totes les característiques disponibles per a cada divisió de node pot ser beneficiós en aquest context específic.

Mirant la taula que pots ser més concís, veiem que els valors del 25 al 50 de max_features són constants i per tant, no ens generen tant interès, ja que podem evitar així sobre ajustaments.

La taula anterior mostra els resultats fins aconseguir el valor més interessant per nosaltres amb tots els seus paràmetres, i el últim valor per mostrar que és molt semblant a diferència dels altres, en la gràfica també es veu representat per molt que hi hagi salts entre valors.

En conclusió, el paràmetre max_features és un paràmetre que va augmentant fins a mantenir una constant amb el specifity i es manté constant amb el f1_score, per tant el valor més òptim és el 25, ja que podem així evitar sobre ajustaments.

Millors valors de cada hiperparàmetre per el model

Un cop hem fet el estudi per els hiperparàmetres més rellevants per el Random Forest i haver extret els valors més òptims (max_features = 25, max_depth = 5 i n_estimators = 200), hem fet una crida al model passant aquest valors per obtenir el millor valor possible de F1 score evitant conflictes amb el dataset.

El resultat:

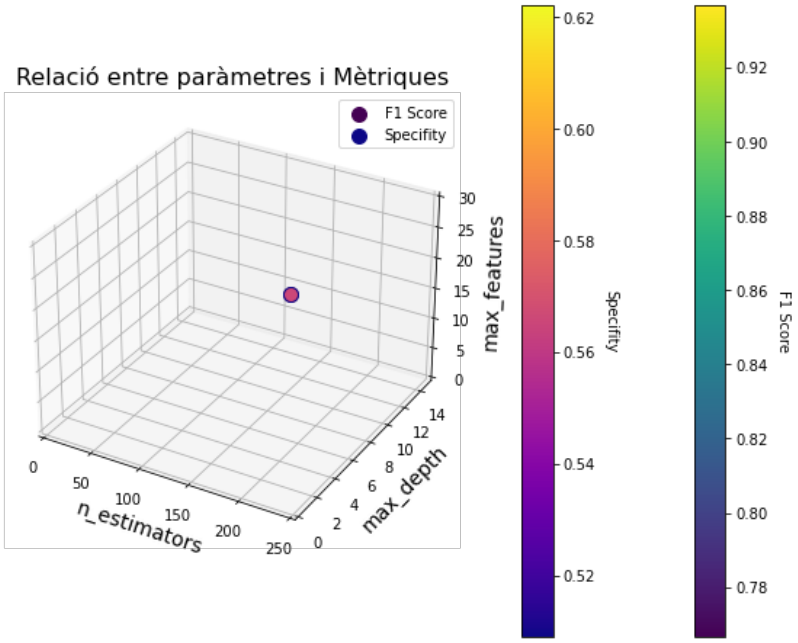
max_depth	max_features	n_estimators	f1_score.	specifity
5.0	25.0	200.0	0.851774	0.565657

Les conclusions que extraiem d'això, son que els valors triat produeixen un F1 score relativament alt 0.851774 i un specifity de 0.565657, és a dir molt interessant veient els primers resultats que no eren superiors a 0.2, el que indica un bon equilibri entre precision i recall, i els valors negatius.

La nostre mètrica particular és important i sembla que aquests valors maximitzen aquesta mètrica especifica. També que els valors moderats de max_depth (5) poden ajudar a evitar el sobre ajustament, especialment en conjunts de datasets petits. Que limitar max_features a 25 podria reduir la variància i evitar models excessivament complexos que podrien sobreajustar-se al conjunt de dades d'entrenament. També que un valor petit de max_depth (5) indica un arbre relativament somni, el que pot ser eficient per als conjunts de dades petits com el nostre. Que com hem dit limitar el max_features a 25 redueix la dimensionalitat i podria millorar l'eficiència del model.

Per seguir, el fet d'utilitzar un n_estimators de 200 proporciona un nombre moderat d'estimadors per al nostra dataset petit, ajudant a evitar sobre ajustament i millorar la generalització. En conjunt de dades petits, reduir la complexitat del model (max_depth i max_features) pot ser essencial per evitar sobre ajustament.

Per concloure, és important saber que els valors òptims poden variar segons el conjunt de dades, la mètrica ii altres factors. Ajustar aquests hiperparàmetres té una part d'art i ciència, i pot ser necessari provar diferents combinacions per trobar la millor configuració. També és important assegurar-se que aquests valors ofereixin un bon rendiment en conjunts de dades nous per garantir la seva generalització.


Relació entre paràmetres i Mètriques
F1 Score
Specifity
n_estimators
max_depth
max_features
Specifity
F1 Score

SVM

Efecte dels hiperparàmetres al model

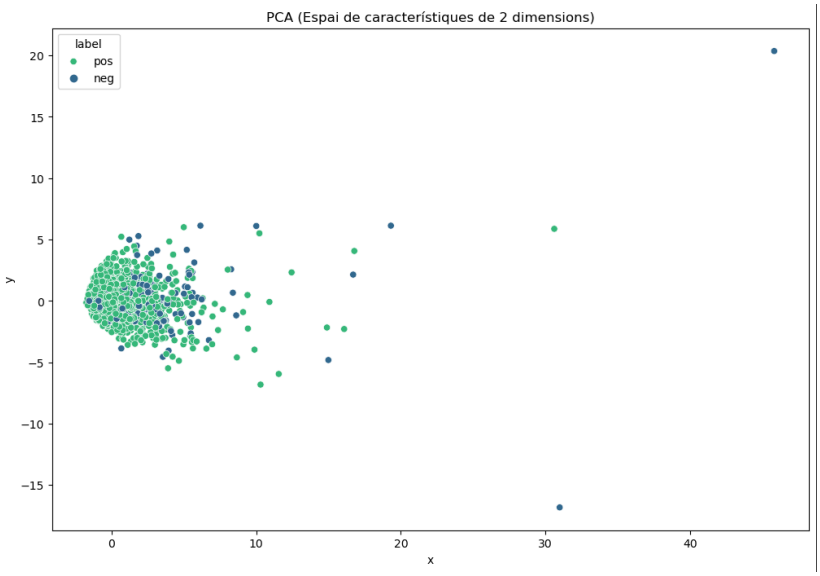
El model SVM té diferents hiperparàmetres ajustables per adaptar el seu rendiment a les nostres dades. Els treballats i comparats al nostre treball seran el paràmetre C, el kernel emprat al model i el paràmetre gamma.

Representació de les dades enfocat al model

En primer lloc cal entendre com estan distribuïdes les dades per poder saber quin serà el millor kernel a utilitzar i seguidament veure com la C i gamma (en cas que no sigui lineal) poden afectar als resultats.

Per aquesta raó partint de la matriu formada amb CountVectors de tamany NxM, on N és el nombre de reviews i M la llargada del vector que conta les paraules, volem aconseguir reduir-la perquè es pugui representar en una gràfica de dues dimensions.
Per fer això hem utilitzat la funció PCA la qual pren les dues característiques (2D) ,escollides perquè explican la major quantitat de la varinça de les dades, i per cada review canvia aquestes caracterísitiques per valors trets apartir d'una combinació lineal de totes les observacions originals.

El resultat és el següent:



El problema d'aquesta representació és que no ens dona informació fiable ja que està omitint a 23 columnes, per tant 23 dimensions.
Igualment, es interesant per poder veure com de complexa és la relació de les dades respecte a la classificació ja que no es pot extreure amb facilitat conclusions ja que no hi ha gaires paraules molt positives o molt negatives.

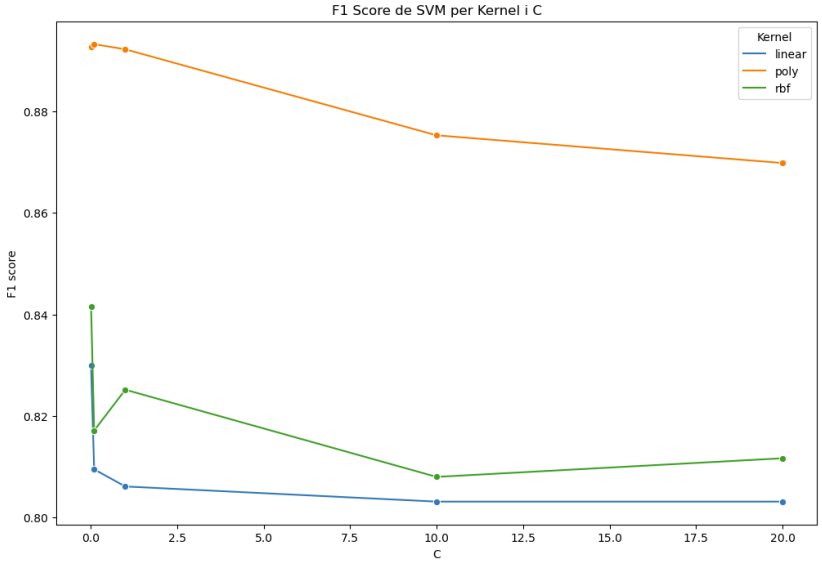
Per aquesta raó ens varem a resultats teòrics respecte els valors resultants de les prediccions i no en deduccions tretes per com estigui representades gràficament les dades.

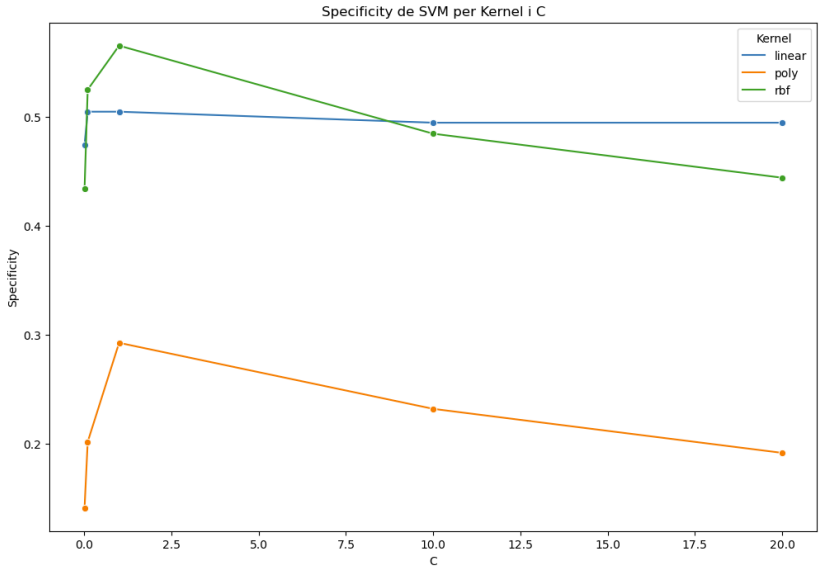
Comparació del paràmetre C amb diferents kernels.

El nostre estudi del paràmetre C va condicionat a l'estudi del millor kernel. La decisió d'un bon kernel es important per permetre al model treballar amb diferents complexitats de dades, per controlar l'overfitting del model a les dades i per la decisió de bons híperparàmetres com pot ser el paràmetre C per això el tractem juntament a l'apartat.

El paràmetre C controla l'equilibri entre maximitzar el marge y minimitzar els errors als punts d'entrenament. Si C augmenta, el model s'ajustarà menys a les dades i per tant reduirà la possibilitat d'aparició d'overfitting però amb el risc de que no predigui bé per no estar suficientment ajustat. Per contra un valor petit de C pot ajustar massa les dades al model generant overfitting.

El kernel que emprem ens pot ajudar a escollir un bon paràmetre C. Primerament amb les següents gràfiques on podem observar la puntuació f1-score i la "specificity".





En aquestes gràfiques podem veure el comportament de cada kernel amb el paràmetre C,

- Poly: Inicialment a la primera gràfica on comparem respecte a f1-score podem veure que el kernel poly es el que millor resultats genera però un cop veiem l'specificity podem observar que la detecció de vertaders negatius es molt baixa respecte a altres kernels això ve produït per la sensibilitat d'aquest kernel al desbalancejament de les dades. La diferencia amb els altres kernels és aquesta sensibilitat que en altres kernels pel disseny intern d'aquests son menys sensibles a aquest problema. El kernel poly intenta trobar una frontera que separi les classes de manera òptima, fins i tot en casos de desbalanceig de dades. Si una classe està subrepresentada, el model es pot acabar centrant massa en aquesta classe i oblidant-se de les instàncies de l'altra classe. Altres kernels com RBF o lineal son menys sensibles a aquest problema degut al disseny intern

per això podem observar que tot i que generant un pitjor f1-score tenen una millor detecció de vertaders negatius.

- Linear i RBF: Es tracten dels kernels més adients a la nostra situació perquè son menys sensibles al desbalanceig produint així que obtinguem resultats d'specificity millors. Basant-nos en les gràfiques podem observar que no son els que millors f1-score obtenen però si que son els que millor relació f1-score/specificity ens proporcionen.

Apartir d'aquests kernels hem fet un petit estudi de l'f1-score obtingut per a cada C amb el kernel seleccionat a cada cas per obtenir la millor C i kernel que serà la base per trobar una gamma adient al treball.

C -> F1-Score

0	linear	0.01	0.829972	10	rbf	0.01	0.841553
1	linear	0.10	0.809485	11	rbf	0.10	0.817133
2	linear	1.00	0.806117	12	rbf	1.00	0.825169
3	linear	10.00	0.803135	13	rbf	10.00	0.808002
4	linear	20.00	0.803135	14	rbf	20.00	0.811667

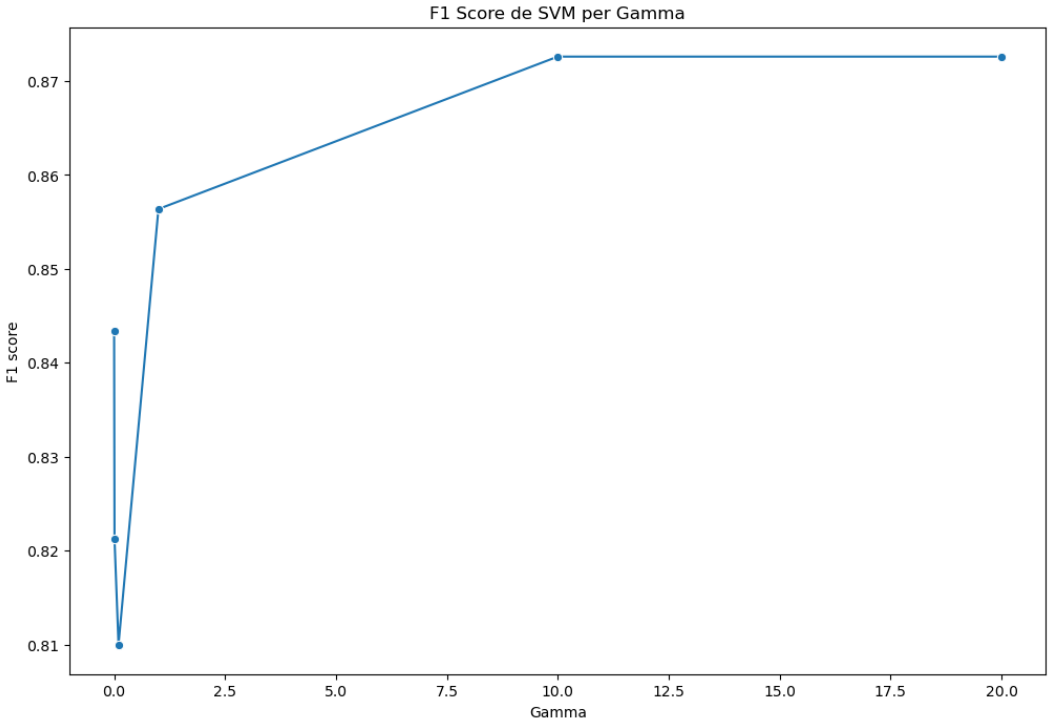
C -> Specificity

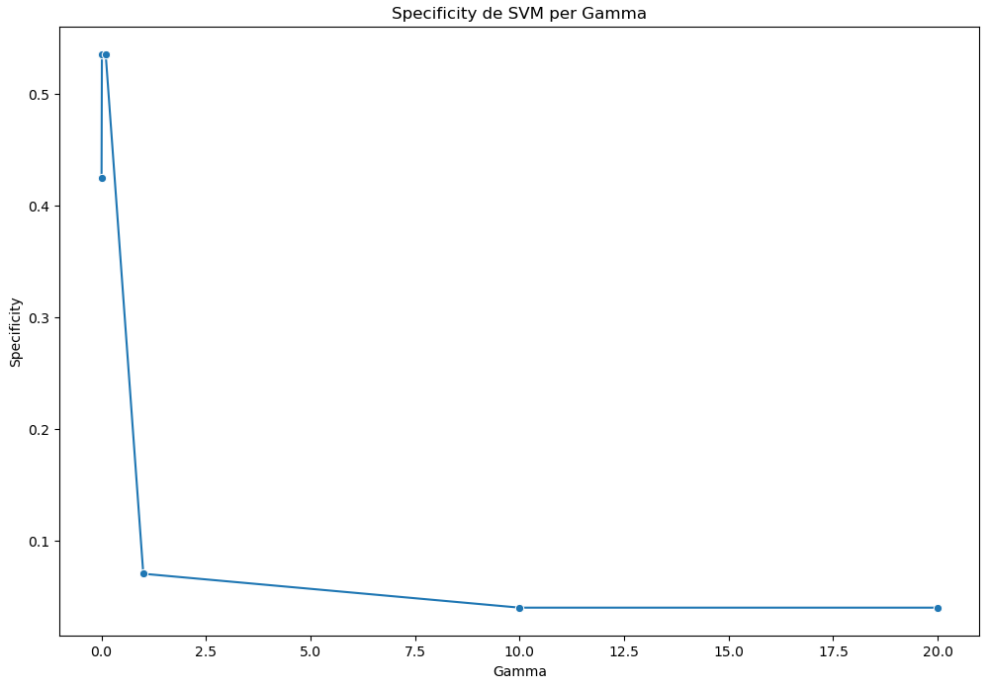
0	linear	0.01	0.474747	10	rbf	0.01	0.434343
1	linear	0.10	0.505051	11	rbf	0.10	0.525253
2	linear	1.00	0.505051	12	rbf	1.00	0.565657
3	linear	10.00	0.494949	13	rbf	10.00	0.484848
4	linear	20.00	0.494949	14	rbf	20.00	0.444444

Apartir de les dades obtingudes podem observar que el millor rendiment el trobem amb el kernel rbf i C = 1.00 que obtenim la relació f1-score/specificity 0.825169/0.565657 respectivament. Aquests seran els valors que emprarem per trobar la gamma més adient.

Paràmetre Gamma

El paràmetre gamma adient s'ha de seleccionar basant-nos en evitar overfitting per evitar així una baixa specificity, gamma es el paràmetre que determina com de similars han de ser les diferents reviews per tal de que es considerin del mateix tipus. Això es traduït a que a mes gran sigui gamma més probabilitat de sobre ajustament tindrà el nostre model reduint així la specificity. Les següents gràfiques/taules mostren l'f1-score i la specificity del model en funció de diferents paràmetres gamma que ens permetran trobar un balanç per trobar el valor de gamma que considerem òptim.





	Gamma	F1 score
0	0.001	0.843448
1	0.010	0.821241
2	0.100	0.810008
3	1.000	0.856372
4	10.000	0.872596
5	20.000	0.872596

	Gamma	Specificity
0	0.001	0.424242
1	0.010	0.535354
2	0.100	0.535354
3	1.000	0.070707
4	10.000	0.040404
5	20.000	0.040404

Podem observar que a més gran sigui gamma com abans s'ha esmentat pitjor specificity obtenim. Per tant podem observar que amb gamma = 0.01 tenim la millor relació f1-score/specificity amb 0.821241/0.535354 respectivament. (Totes aquestes mesures son amb el kernel RBF i C=1.00).

Comparació i conclusió de models

El nostre treball ha a comparat els models i ha estudiat els hiperparàmetres a partir de la puntuació f1-score i la specificity. A continuació mostrem els valors d'aquestes mesures mesurats amb els millors hiperparàmetres segons els nostres estudis.

MODEL	**PARÀMETRES**	**F1-SCORE**	**SPECIFICITY**
LOGISTIC REG.	-*	0.796549	0.545454
RAND. FOREST	max_features = 25 ; max_depth = 5 ; n_estimators = 200	0.851774	0.565657
SVM	C= 1.00 ; Kernel = RBF ; Gamma = 0.01	0.821241	0.535354

** En Logistic Regression no s'ha modificat cap paràmetre, sinó que s'ha fet servir com a model per veure com afecta la quantitat de dades.*

Com a conclusió podem observar que el model que millors mètriques d'avaluació ha obtingut ha estat el Random Forest amb la la relació f1-score/specificity = 0.851774/0.565657. Tot i aquest ser el millor no dista molt de SVM.

Pas a pas del projecte