Projet Data Mining

Etude de la base de données *Onlinenewspopularity*

Léa BRASSEUR & Benjamin CHAUVET

Table des matières

1 Introduction

1.1 Objectifs

Notre base de données est composée de **61** variables sur **39644** articles publiés sur le site Mashable. Au premier semestre, nous avions étudié ce qui influençait le nombre de partages des articles. Nous avions conclu que certains thèmes d'articles sont plus partagés et que les articles publiés le weekend sont plus populaires. Tout comme certaines caractéristiques sur le contenu des articles impactent le nombre de partages.

Dans cette étude, nous allons déterminer le meilleur modèle prédictif de notre base de données. L'objectif étant de prédire selon les caractéristiques d'un article s'il sera populaire ou non.

1.2 Nettoyage et recodage de la base de données

Tout d'abord, nous avons créé un autre *dataframe* basé sur le premier, avec les variables **url** et **timedelta** en moins car elles étaient non prédictives. Nous avons également supprimé les articles avec 0 mots car nous ne trouvions pas cela pertinent dans notre analyse.

Puis, nous avons transformé nos variables quantitatives binaires sur les thèmes des articles et les jours de publication de la semaine en qualitatives.

Par la suite, nous avons crée notre variable à prédire **Popularity** à partir du nombre de partages.

Le but de nos analyses étant donc de prédire l'appartenance des différents articles à la classe *Popular* ou *Unpopular* en fonction des caractéristiques de ceux-ci. Nous avons choisi cette séparation de la variable de popularité autour de la médiane pour obtenir deux classes avec des effectifs homogènes.

Table 1 – Répartition de la variable popularity

Popularity	Effectif
Popular	18912
Unpopular	19551

1.3 Echantillonage

Nous avons constaté que la variable **n_non_stop_words** est constante à l'intérieur des différents groupes, nous sommes donc obligés de la supprimer pour que les modèles suivants fonctionnent. De plus, nous avons supprimé la variable **shares** sur le nombre de partages car celle nous a permis de créer la variable de popularité.

Nous séparons ensuite notre base de données en deux échantillons distincts : l'un d'apprentissage et l'autre de test. Nous avons choisi une proportion de 2/3 pour notre échantillon d'apprentissage car notre de base de données étant assez fournie, nous avions grâce à ce découpage des échantillons d'apprentissage et de test assez importants pour pouvoir créer et tester nos modèles.

Table 2 – Répartition des échantillons

Apprentissage	Test
25642	12821

2 Analyses factorielles discriminantes

2.1 Linear discriminant analysis

Nous avons créé notre modèle dans le but de prédire l'appartenance des articles aux classes de popularité grâce aux **46** variables présentes dans notre base de données. Puis, par la suite nous avons appliqué notre échantillon test à ce modèle.

Pour mesurer la qualité du modèle nous avons créé une matrice de confusion. Et cette matrice nous a ensuite permis de calculer les différentes erreurs ayant été faites au cours de la prédiction. Par la suite, nous avons pu faire les courbes ROC associées à ces erreurs.

TABLE 3 – Matrice de confusion

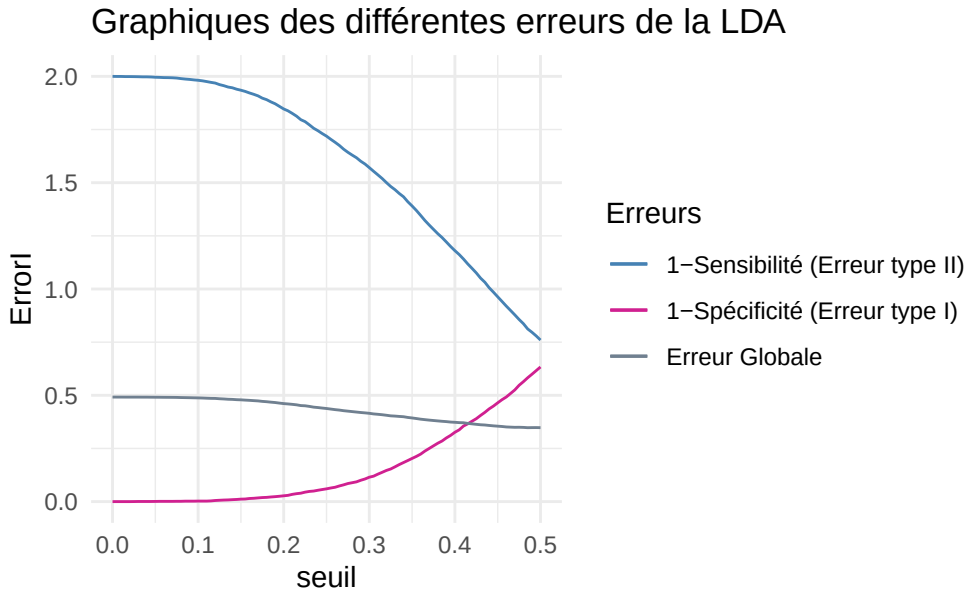
	Réalité		
	Popular	Unpopular	Sum
Popular	3909	2064	5973
Unpopular	2395	4453	6848
Sum	6304	6517	12821

Puis, nous avons fait la division des prédictions par la somme des valeurs de la réalité.

TABLE 4 – Proportion en fonction de la réalité

	Réalité	
	Popular	Unpopular
Popular	0.62	0.317
Unpopular	0.38	0.683

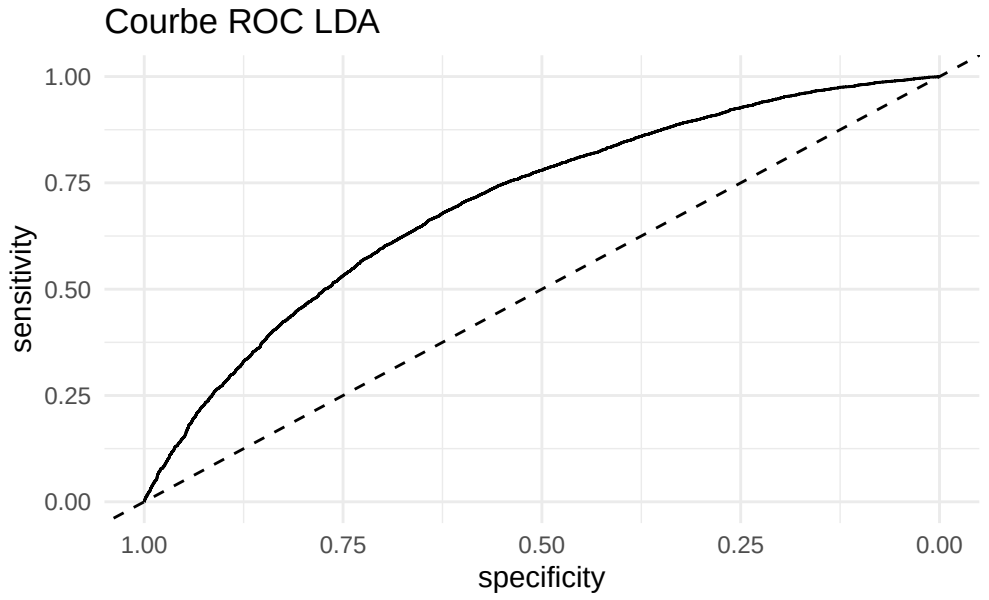
— L'erreur globale est de 0.348. C'est-à-dire que 34.8% de la base de données est mal estimé.
— L'erreur de type I est de 0.317 et correspond à 1 - la spécificité. Il s'agit du taux de faux positifs, c'est-à-dire la part d'articles prédits populaires alors qu'ils sont en réalité impopulaires. La spécificité étant la capacité à identifier les "vrais" articles impopulaires.
— L'erreur de type II est de 0.38 et correspond à 1 - la sensibilité. Il s'agit du taux de faux negatifs, c'est-à-dire la part d'articles prédits impopulaires alors qu'ils sont en réalité populaires. La sensibilité étant la capacité à identifier les "vrais" articles populaires.



Grâce à ce graphique nous pouvons voir que l'erreur de type II est supérieure aux autres, ce qui signifie que notre modèle a plus de mal à prédire les articles étant populaires.

La matrice de confusion nous permet également de calculer l'accuracy qui est de : 0.652. Ce qui veut dire que 65.2% de la base de données est bien prédite.

Nous pouvons par la suite créer une courbe ROC grâce à la spécificité et à la sensibilité.



Ici notre AUC, qui représente l'aire sous la courbe est de 0.702. Elle est plus élevée que 0.5 ce qui veut dire que nous arrivons à collecter une information, mais elle est relativement inférieure à 1, les prédictions ne sont donc pas optimales.

2.2 Quadratic discriminant analysis

Nous avons créé une deuxième base de données, toujours en partant de la base de données initiale. Dans celle-ci nous avons été obligés de supprimer la variable constante **n_non_stop_words**. Et également d'enlever les variables **weekday_is_friday**, **is_weekend** et **LDA_04** pour cause de colinéarité.

Nous séparons ensuite notre base de données en deux échantillons, avec toujours une proportion de 2/3 pour celui d'apprentissage. Puis nous effectuons notre QDA.

Le modèle nous donne la matrice de confusion ci-dessous.

Table 5 – Matrice de confusion

	Réalité		
	Popular	Unpopular	Sum
Popular	1590	606	2196
Unpopular	4714	5911	10625
Sum	6304	6517	12821

Puis le tableaux des proportions en fonction de la réalité.

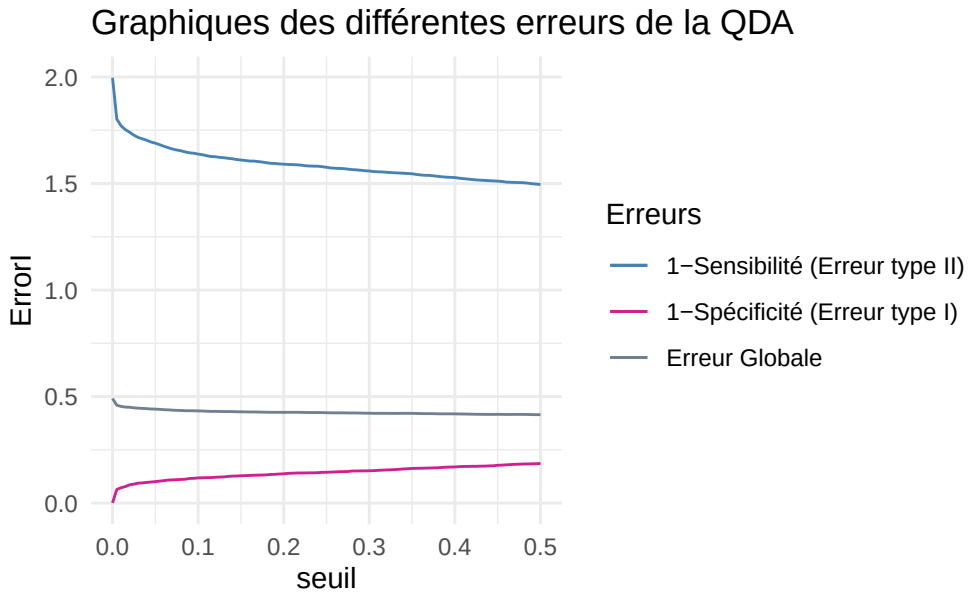
Table 6 – Matrice des proportions en fonction de la réalité

	Réalité	
	Popular	Unpopular
Popular	0.252	0.093
Unpopular	0.748	0.907

Nous remarquons que :

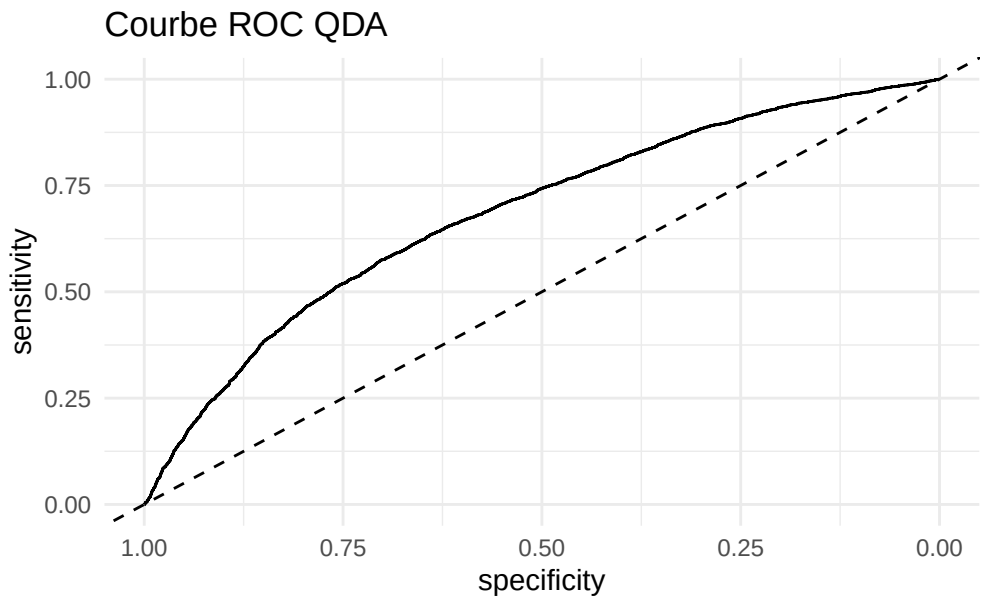
- Notre modèle a une erreur globale de 41.5% ce qui est plus que pour la LDA et donc moins performant.
  - L'erreur de type I est de 9.3%, bien moins que celle en LDA.
  - Cependant, l'erreur de type II est de 74.8% ce qui correspond au taux de faux négatifs, c'est-à-dire l'erreur d'affectation d'articles populaires qui sont prédits impopulaires.
- Le taux de bonnes affectations est de 58.5%.
  - Seulement 25.2% des articles populaires sont biens prédits.
  - 90.7% des articles impopulaires sont biens prédits.

Le modèle QDA est plus performant pour prédire si un article va être impopulaire alors qu'il se trompe plus pour prédire si un article sera populaire.



Ce graphique nous montre que l'erreur de type II est beaucoup plus élevée que l'erreur globale ou que l'erreur de type I. Ces conclusions sont en accord avec celles faites au dessus comme quoi la QDA prédit mieux les articles impopulaires car ceux-ci sont représentés par l'erreur de type I.

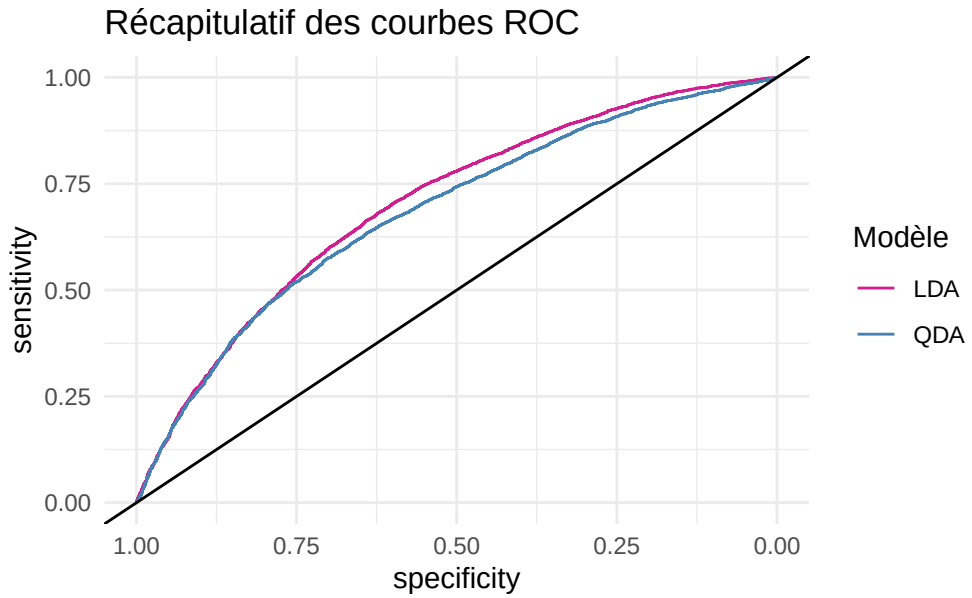
Nous avons ensuite tracé la courbe ROC allant avec ce modèle.



La courbe ROC reste relativement proche de la bissetrice tout comme pour la LDA. De plus, l'aire sous la courbe est de 0.683.

2.3 Comparaison LDA et QDA

Nous pouvons faire une rapide comparaison entre les deux modèles faits jusqu'ici, la LDA et la QDA.



Nous pouvons voir grâce au graphique que la différence entre les deux modèles est minime, même si la LDA semble légèrement plus performantes puisque la courbe est plus éloignée de la bissectrice.

3 K plus proches voisins

Pour effectuer notre modèle sur les K plus proches voisins, nous reprennons notre échantillon utilisé lors de la QDA (avec les 3 variables qui posent un problème de colinéarité en moins). Nous testons en premier lieu un modèle des KNN avec un nombre de voisins de 6 (racine carrée de notre nombre de variables).

Table 7 – Matrice de confusion

	Réalité		
	Popular	Unpopular	Sum
Popular	3030	3043	6073
Unpopular	3274	3474	6748
Sum	6304	6517	12821

— Nous obtenons une erreur globale de 49.3% ce qui n'est pas très performant comparé à nos modèles précédents.

Afin d'améliorer ce modèle des K plus proches voisins, nous allons déterminer le nombre de voisins optimal minimisant l'erreur d'entrainement. Pour cela nous faisons une optimisation de nos KNN à l'aide d'une validation croisée.

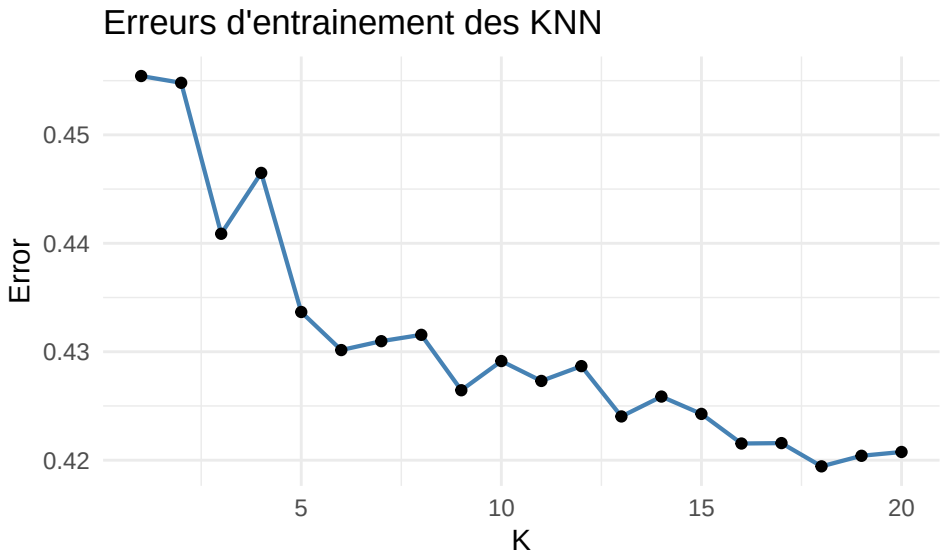


Table 8 – Meilleur nombre de voisins

	K	Erreur d'entrainement
18	18	41.943

Notre nombre optimal de voisins est de 18. Nous allons donc refaire notre modèle en partant avec cette fois-ci 18 centres initiaux. Nous obtenons cette matrice de confusion.

Table 9 – Matrice de confusion

	Réalité		
	Popular	Unpopular	Sum
Popular	3291	2395	5686
Unpopular	3013	4122	7135
Sum	6304	6517	12821

— Notre erreur globale est maintenant de 42.2%. Comme prévu, ce modèle est plus performant que le précédent grâce à l'optimisation du nombre de centre. Même s'il reste moins efficace que la LDA ou la QDA.

Table 10 – Sommaire

	x
Sensitivity	0.5220495
Specificity	0.6324996
Pos Pred Value	0.5787900
Neg Pred Value	0.5777155
Precision	0.5787900
Recall	0.5220495
F1	0.5489575
Prevalence	0.4916933
Detection Rate	0.2566882
Detection Prevalence	0.4434911
Balanced Accuracy	0.5772746

Grâce à ce résumé nous pouvons voir que la sensibilité et la spécificité ne sont pas très hautes ce qui n'est pas bon. Cela veut dire que les articles qu'ils soient populaires ou impopulaires sont mal identifiés dans notre base de données. Nous notons quand même qu'encore une fois la spécificité est plus importante, ce qui veut dire que les articles impopulaires sont légèrement mieux prédits. De plus, l'accuracy ici a une valeur très basse, notre modèle de KNN, même amélioré identifie et prédit très mal nos données.

4 Arbres de classification

4.1 Arbres de décisions

Dans cette partie, nous avons utilisé le même *dataframe* que pour la QDA. Dans un premier temps nous avons fait un arbre de décisions avec toutes nos variables (46), cependant celui-ci n'était pas lisible car notre base de données est trop volumineuse.

Mais nous avons quand même pu faire une matrice de confusion et une matrice de proportions en fonction de la réalité.

Table 11 – Matrice de confusion

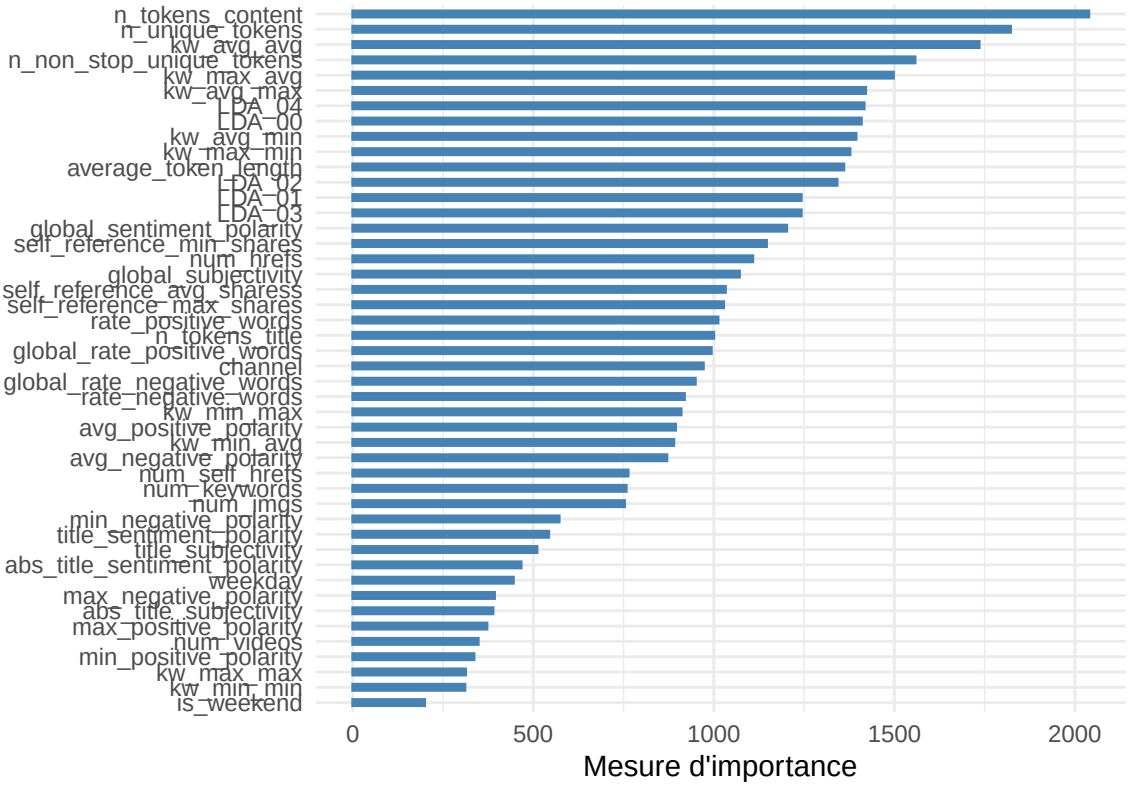
	Réalité		
	Popular	Unpopular	Sum
Popular	3667	2798	6465
Unpopular	2637	3719	6356
Sum	6304	6517	12821

Table 12 – Proportion en fonction de la réalité

	Réalité	
	Popular	Unpopular
Popular	0.567	0.433
Unpopular	0.415	0.585

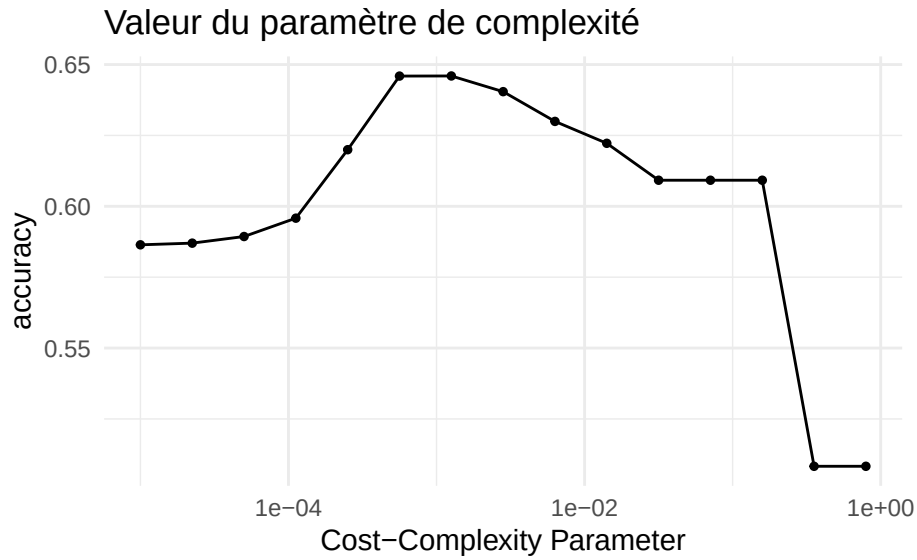
— Nous observons une erreur globale de 42.4%.
— Nous avons 56.7% d'articles populaires et 58.5% d'articles impopulaires qui sont bien prédits.

Nous pouvons aussi afficher l'importance des variables.



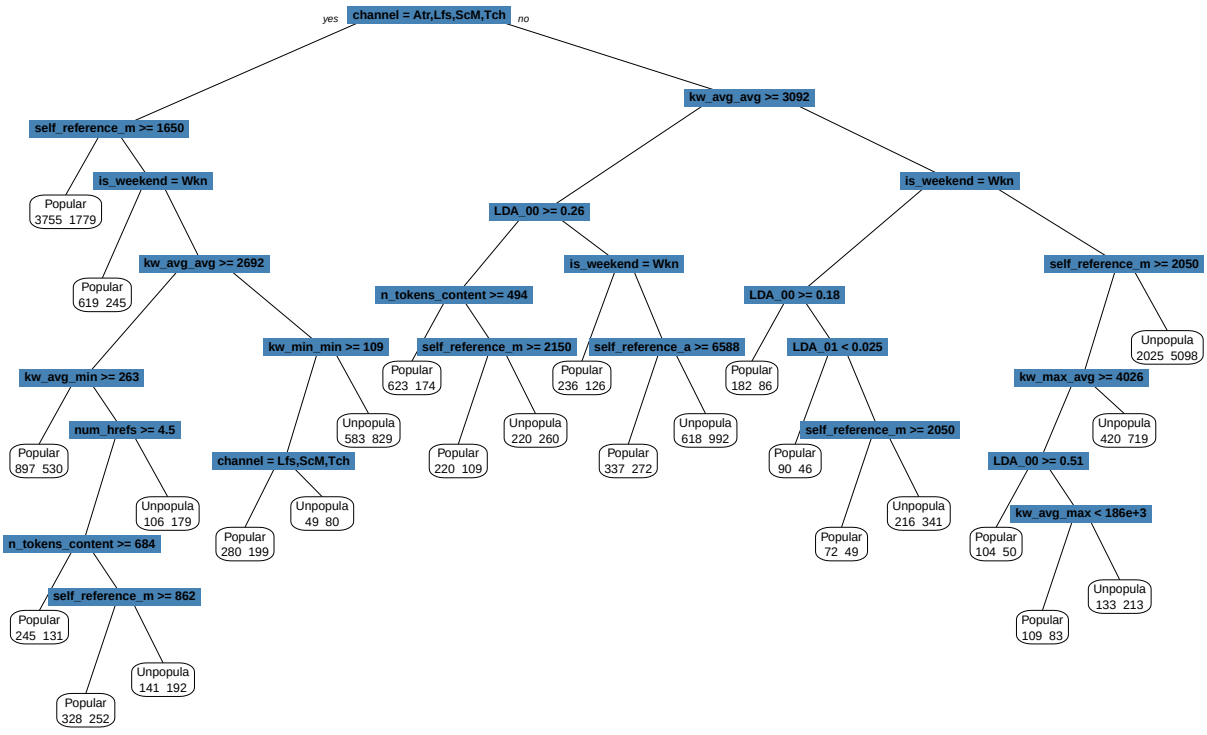
Ce graphique nous montre que les variables les plus importantes ici sont **n_tokens_content**, **n_unique_tokens** et **kw_avg_avg**. Ces variables sont en rapport avec les textes des articles. Nous pouvons donc supposer que ce modèle se base sur le contenu des articles pour pouvoir les affecter à une catégorie (*popular* ou *unpopular*).

Par la suite, pour pouvoir avoir un arbre lisible, nous avons décidé d'en construire un nouveau en changeant le paramètre de complexité. Pour cela, nous avons utilisé tidymodels pour pouvoir optimiser celui-ci.

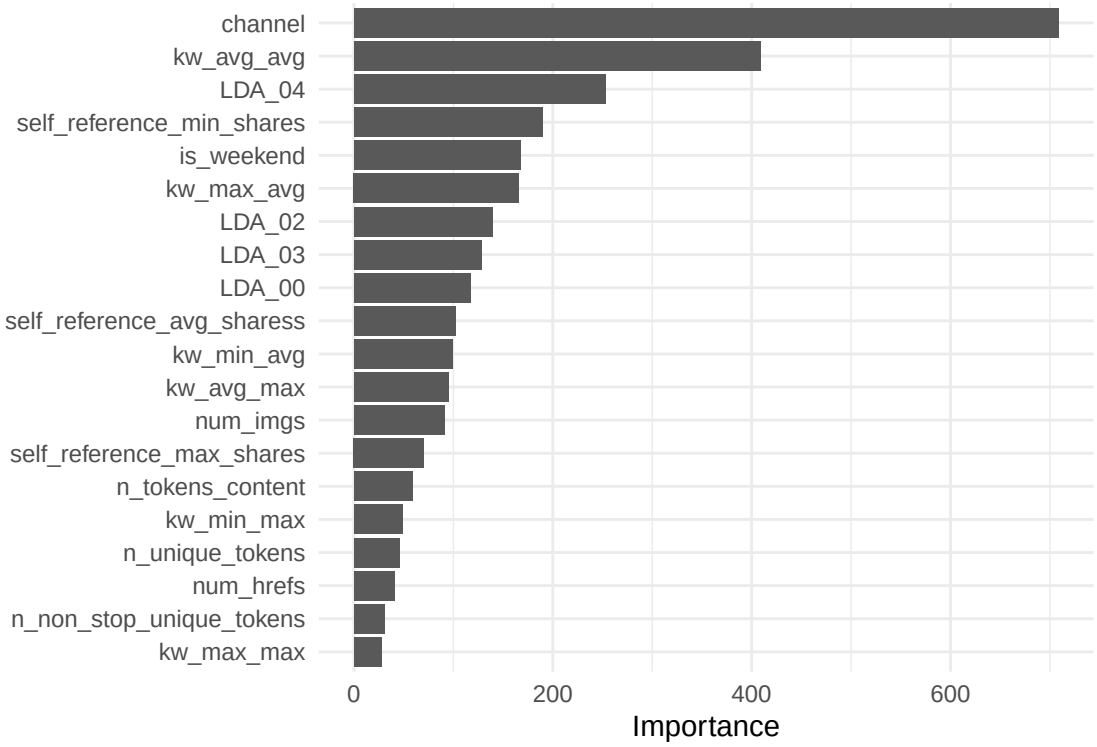


Ce graphique nous donne les valeurs de l'accuracy en fonction de différentes valeurs pour le paramètre de complexité. La valeur du paramètre de complexité qui maximise l'accuracy est de 0.00126.

Nous avons donc pu recréer un arbre cette fois-ci avec le paramètre de complexité optimal. Voici la version élagué de notre arbre.



Pour pouvoir mieux distinguer les variables qui sont importantes dans notre modèle il est possible de faire un diagramme en bâtons. Nous pouvons donc voir que cette fois-ci ce ne sont plus les mêmes variables qui interviennent dans le modèle. Effectivement, la variable **channel** à une importance capital dans la répartition entre populaire et impopulaire, puis viennent **kw_avg_avg** qui se rapportent au mots-clés dans l'article et enfin **LDA_04** qui à un lien avec les thèmes.



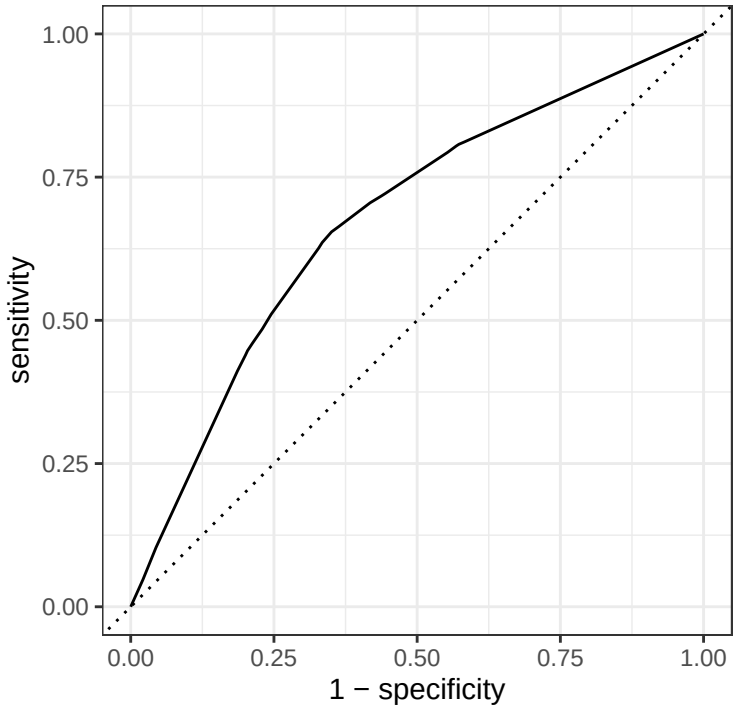
Puis la matrice de confusion de ce modèle.

Table 13 – Matrice de confusion

	Réalité	
	Popular	Unpopular
Popular	4126	2285
Unpopular	2178	4232

— Ce nouveau modèle a une erreur globale de 34.8%.
— Nous avons également 64.4% d'articles populaires et 66% d'articles impopulaires qui sont bien prédits. Cette fois-ci, il y a peu de différence entre la prédictions des articles de nos deux classes.

Et pour terminer la courbe ROC du modèle.



Le tableau ci-dessous nous montre l'accuracy et l'AUC du modèle élagué, avec le meilleur paramètre de complexité.

Table 14 – Accuracy et AUC

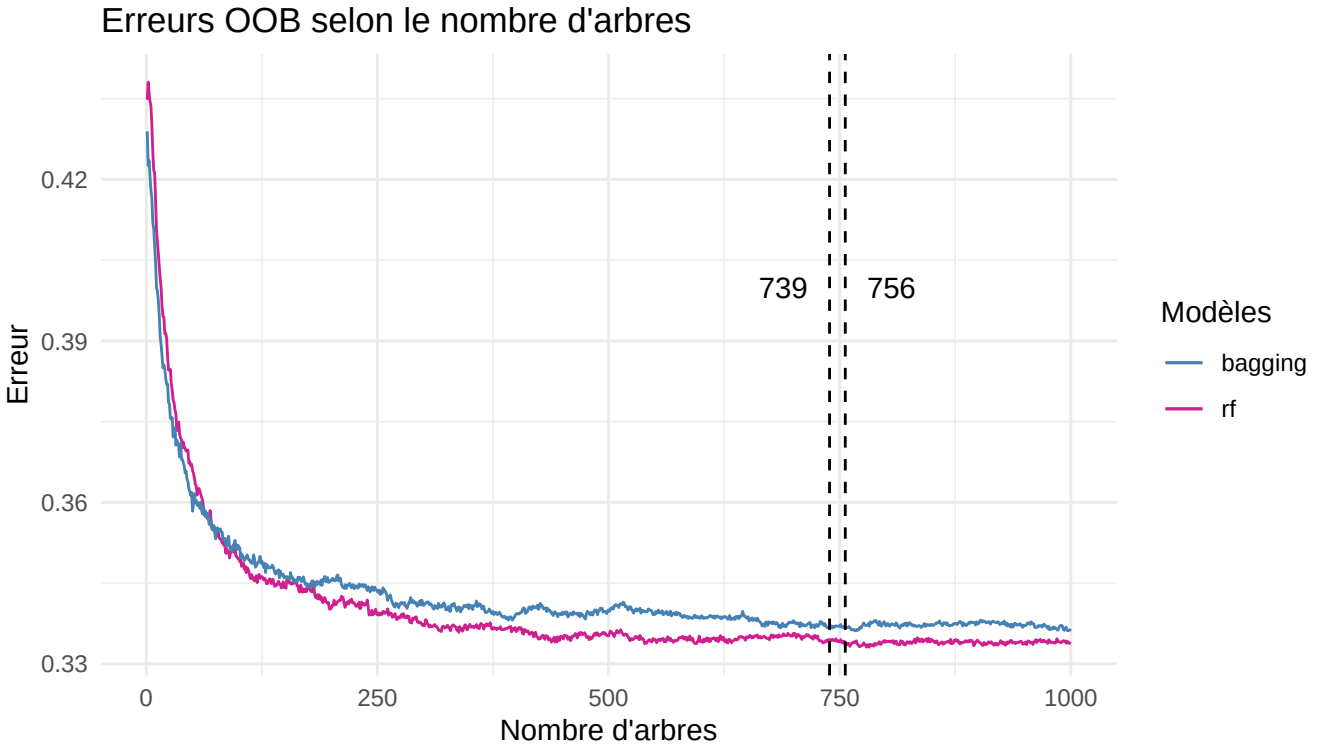
.metric	.estimator	.estimate	.config
accuracy	binary	0.6409796	Preprocessor1_Model1
roc_auc	binary	0.6777138	Preprocessor1_Model1

4.2 Forêts aléatoires et bagging

4.2.1 Nombre d'arbres

La base de données utilisée ici est la même que pour la LDA et les arbres de décisions. Dans un premier temps, regardons ce que donne un modèle de forêt aléatoire et de bagging en terme d'erreur Out-Of-Bag. Nous cherchons donc le nombre d'arbres qui minimise cette erreur.

Pour le bagging, il s'agit d'un cas particulier de forêt aléatoire construit avec tous les prédicteurs (46 variables).



Nous remarquons que la courbe du modèle de forêt aléatoire est légèrement plus basse que celle du bagging. Le nombre d'arbres optimal qui minimise l'erreur Out-Of-Bag pour le modèle de forêt aléatoire est de 756 arbres alors qu'on obtient 739 arbres pour le bagging. Au-delà, l'erreur des deux modèles ne s'améliore pas car les courbes restent constantes.

Cependant, nous obtenons sensiblement les mêmes erreurs autour de 450 arbres ce qui est moins coûteux en terme de ressources.

Comparons ensuite ce que nous donne ces modèles avec 450 arbres contre les modèles avec le nombre d'arbres optimal.

TABLE 15 – Erreurs random forest vs bagging

	Erreur test (ntree optimal)	Erreur test (ntree = 450)
Bagging	0.3404	0.3413
Random forest	0.3405	0.3355

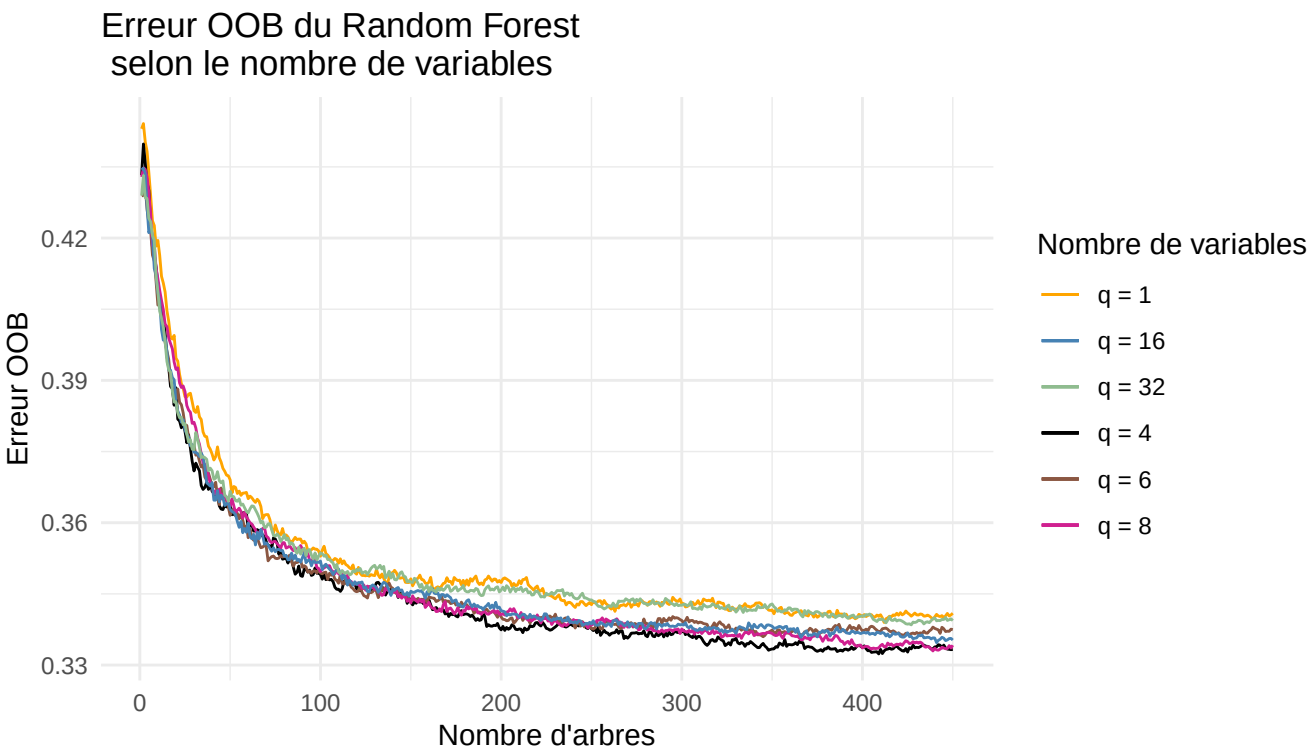
Le meilleur modèle de bagging est celui composé de 739 arbres, il cumule sur nos données de test une erreur globale de 34.04% mais reste très proche de l'erreur test avec 450 arbres qui est de 34.13%.

Ces erreurs associées au bagging restent moins bonnes que les erreurs du modèle de forêt aléatoire qui sont de :

— 34.05% d'erreur pour 756 arbres
— 33.55% pour 450 arbres.

4.2.2 Nombre de variables

Nous allons maintenant regarder quel serait le meilleur nombre de variable à selectionner pour une forêt aléatoire de 450 arbres.



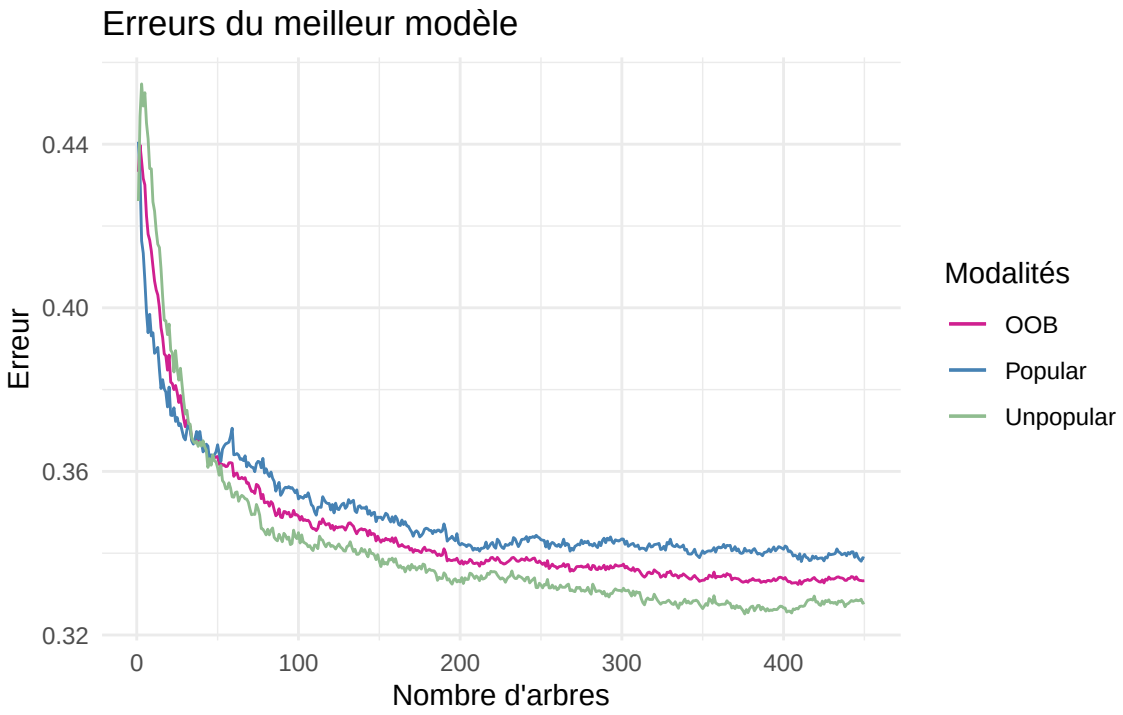
Ces différences d'erreurs bien que très minimes montrent que le modèle avec 4 variables est le plus otpimal pour 450 arbres. C'est d'autant plus le cas que notre échantillon d'entrainement contient relativement beaucoup d'articles (25642 individus). Le tableau récapitulatif suivant confirme les conclusions tirées du graphique.

TABLE 16 – Erreurs selon le nombre de variables

	Type d'erreur	
	Err_test	Err_OOB
RF : q=1	0.3425	0.3390
RF : q=4	0.3416	0.3323
RF : q=6	0.3357	0.3360
RF : q=8	0.3394	0.3330
RF : q=16	0.3403	0.3347
RF : q=32	0.3385	0.3385

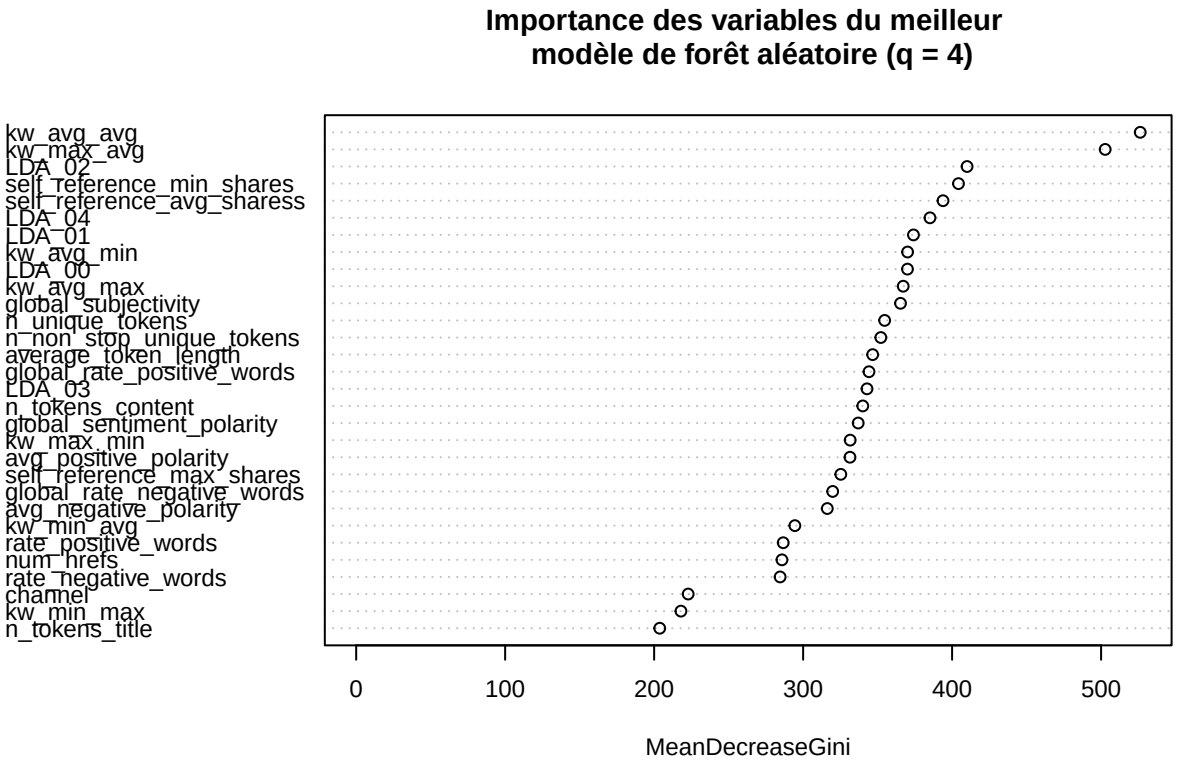
Les erreurs sur nos données test étant très semblables, nous gardons le modèle de forêt aléatoire avec 4 variables (q = 4) car il présente la plus faible erreur Out-Of-Bag qui est de 0.332.

En ce qui concerne la profondeur de nos arbres, les résultats en terme d'erreur n'étaient pas tellement impactés donc nous avons conservé le paramètre par défaut.



Nous remarquons que l'erreur du groupe *Unpopular* est la plus basse, comme pour quasiment tous nos modèles précédents. On en conclut qu'avec notre meilleur modèle de forêt aléatoire, les articles impopulaires sont un peu mieux prédits que les articles populaires. Au final, notre meilleur modèle nous donne une erreur globale de 0.342

4.2.3 Importance des variables

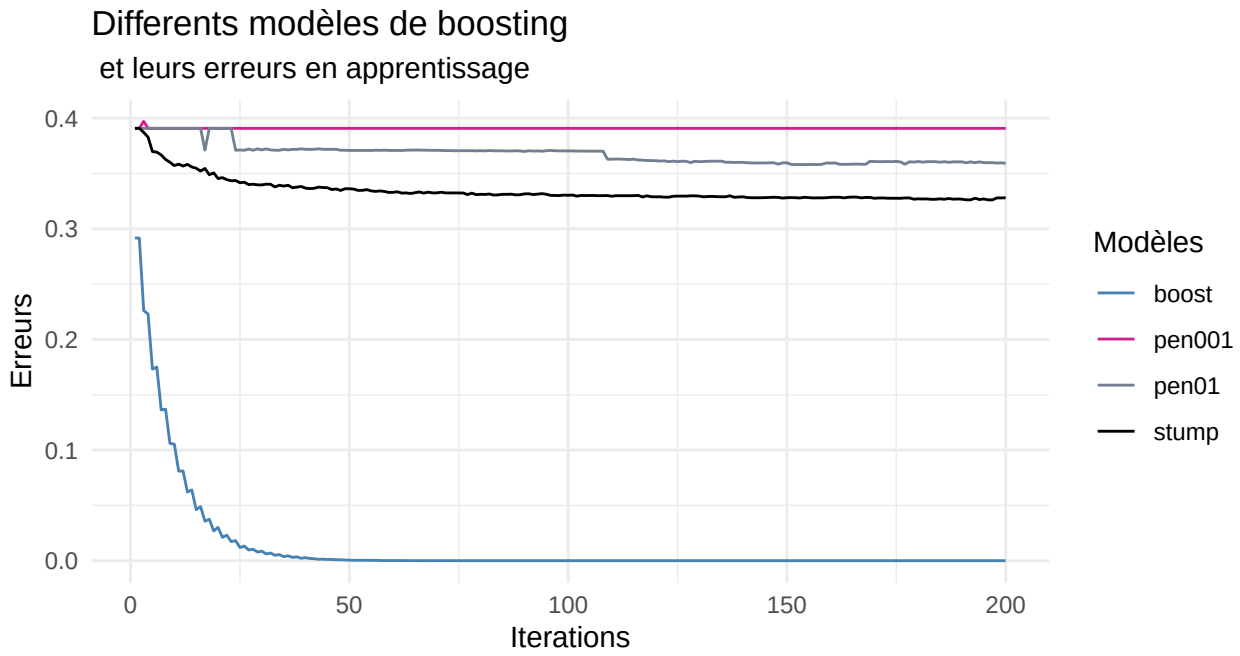


Nous retrouvons avec ce graphique les variables qui jouent un rôle important dans notre meilleur modèle

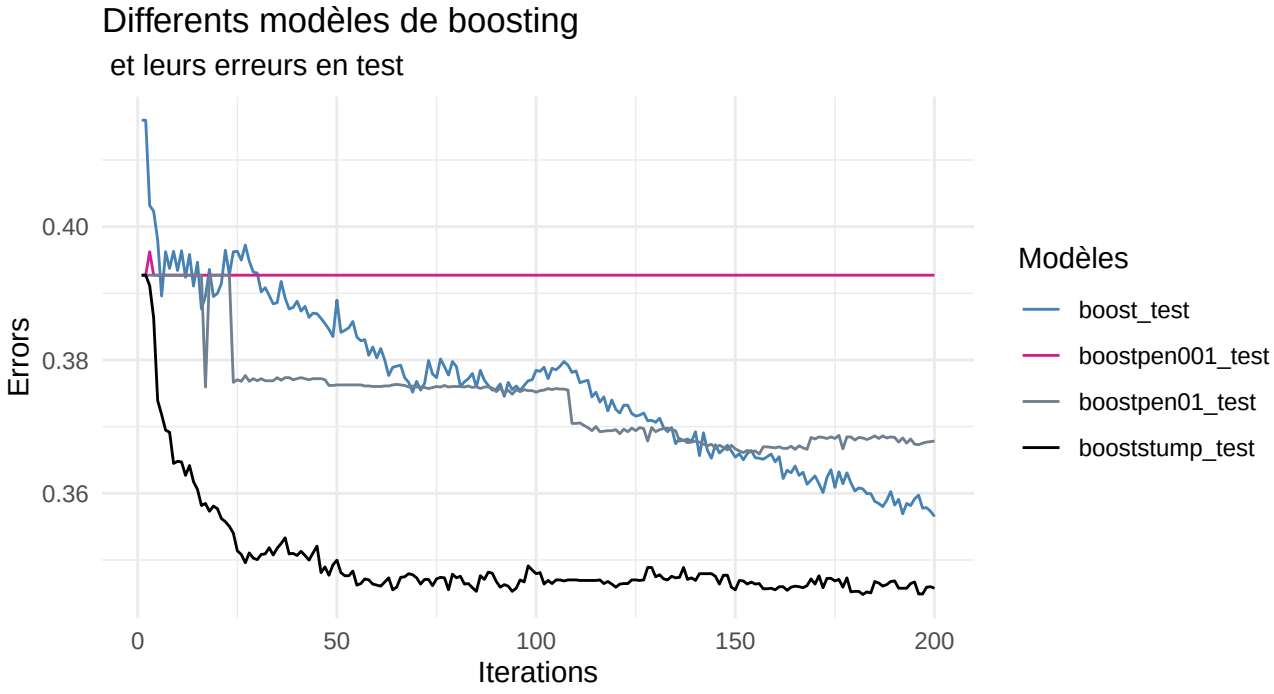
de forêt aléatoire avec 4 variables. On retrouve parmi ces variables **kw_avg_avg** et **kw_max_avg** qui sont celles qui influencent le plus notre modèle et qui se rapportent au mots-clés pouvant se trouver dans les articles. Après, il y a **self_reference_min_shares** ou encore les variables **LDA_02**, **LDA_04** et **LDA_00** qui représenteraient des thèmes des articles.

4.3 Boosting

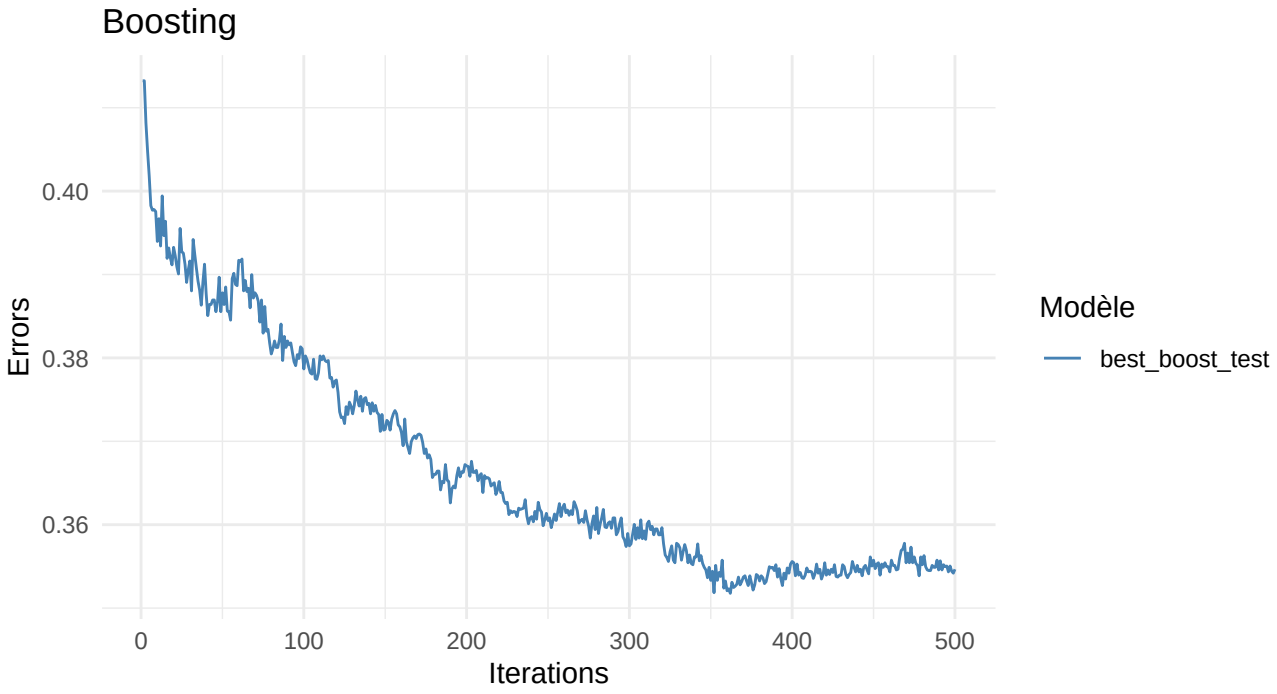
Dans ces derniers modèles la base de données utilisées est la même que pour la LDA. Le but de cette partie va être de comparer différents modèles de boosting. Le premier se fera sur arbre complet, le deuxième sera un boosting avec stumps et enfin deux modèles de boosting avec une pénalisation de 0.01 et de 0.001.



Nous constatons que l'erreur sur les données d'entrainement du boosting chute très rapidement à zéro et que le modèle stump se stabilise en terme d'erreur à partir d'environ 50 itérations. Les modèles de boosting avec pénalisation semblent avoir une erreur constante selon le nombre d'itération.



Grâce à ce grpahique nous pouvons tout de suite voir que le modèle avec les erreurs en test les plus basses est le modèle avec les stumps. Nous remarquons aussi que l'erreur test du boosting semble continuer de descendre. Nous avons donc regardé cette erreur sur 500 d'itérations. Nous n'avons pas fait 500 iterations dès le début car cela est très couteux pour l'ordinateur de plus cela n'aurait pas servi pour les modèles avec pénalisation et stumps car ils semblent se stabiliser avant les 200 iterations.



Ce graphique nous permet de voir que le boosting était le plus performant à 350 itérations environ, au delà l'erreur remontait. Nous pouvons supposer la présence d'un phénomène de surapprentissage.

Table 17 – Les erreurs en test des modèles

	50	100	200	350	400
Boosting	0.388	0.379	0.367	0.353	0.356
Boost Pen 10%	0.376	0.375	0.368		
Boost Pen 1%	0.393	0.393	0.393		
Stump	0.35	0.348	0.346		

Ce tableau confirme les observations faites auparavant grâce au graphique. Le meilleur modèle est celui avec les stumps et 200 itérations. Ceci était également visible grâce aux graphiques puisque sur le premier graphique avec les erreurs test nous pouvons voir que les erreurs pour le modèle stumps descendent en dessous de 0.35. Or, sur le graphique avec uniquement les erreurs du boosting la courbe ne va pas en dessous de 0.35.

Nous avons fait la matrice de confusion associée au modèle stumps.

Table 18 – Matrice de confusion

	Réalité		
	Popular	Unpopular	Sum
Popular	4015	2144	6159
Unpopular	2289	4373	6662
Sum	6304	6517	12821

— L'erreur globale de ce modèle est de 0.346.

— L'erreur de type I est de 0.329, il y a donc 32.9% d'articles qui sont prédits comme étant populaires alors qu'en fait ils sont impopulaires.
— L'erreur de type II est de 0.363, il y a donc 36.3% d'articles qui sont prédits comme étant impopulaires alors qu'en fait ils sont populaires.
— L'accuracy est de 0.654.

5 Comparaison des modèles et conclusion

Pour faire la comparaison entre nos modèles nous allons nous baser sur les erreurs globales et donc sur l'accuracy. Pour se faire nous allons choisir nos meilleurs modèles pour chaque partie, c'est-à-dire : la LDA, la QDA, les KNN avec 18 voisins, l'arbre élagué, la forêt aléatoire avec 4 variables et enfin le boosting avec stumps et 200 itérations.

Table 19 – Conclusion

	Accuracy	Erreur globale	Faux populaires	Faux impopulaires
LDA	0.652	0.348	0.317	0.380
QDA	0.585	0.415	0.093	0.748
KNN	0.578	0.422	0.368	0.478
Tree	0.652	0.348	0.345	0.351
Random forest	0.658	0.342	0.354	0.330
Booststump	0.654	0.346	0.329	0.363

Nous pouvons conclure que notre meilleur modèle est le modèle de forêt aléatoire avec 4 variables et 450 arbres car il a la plus basse erreur sur les données test et donc la meilleure performance (65.8% d'accuray). Cependant, dans une optique de prédire si un article va être impopulaire, le modèle QDA s'avère être le plus performant du fait de son erreur de type I la plus faible. En revanche, l'enjeu de prédire la popularité des articles va plutôt être de prédire si un article donné va être populaire. Ainsi, notre meilleur modèle de forêt aléatoire est le plus adapté du fait de son erreur de type II (0.33) et de test (0.342) qui sont les plus petites en comparaison avec tous les autres modèles.

Nous avons aussi pu remarquer que les articles impopulaires étaient en général mieux identifés et donc mieux prédits dans la quasi-totalité de nos modèles. Cela peut s'expliquer par la répartiton difforme du groupe *Popular* par rapport au groupe *Unpopular*. Pour rappel, ces classes sont séparées à la médiane du nombre de partages qui est de 1400 partages, sauf que le nombre de partages maximal est de 843300 partages. La classe des articles populaires est donc très étendue par rapport à celle des articles impopulaires.

Nos modèles pourraient être améliorés en effectuant une ACM au préalable sur nos données afin de réduire le bruit et donc espérer une meilleure performance de prédiction. Choisir une plus grande proportion de données d'entrainement peut aussi être une solution, nous pourrons proposer 3/4. Ou encore effectuer ces prédictions sur 3 modalités au lieu de 2 pour avoir une classe dédiée aux valeurs extrêmes que représentent les articles les plus partagés.