

Projet d'étude - Analyse de la Supply Chain :

Analyse et prévision des avis laissés sur les plateformes TrustPilot et TrustedShop à partir des commentaires laissés par les clients/internautes

Cohorte de novembre 2022 - parcours Data Scientist

**Adnane MOUZAOU
François ROUXELIN**

Rapport d'étude n°2 :

Rapport de modélisation



DataScientest.com

Agrément organisme de formation 11755665975

09 80 80 79 49

2 place de Barcelone, 75016 Paris

Sommaire

Rappel des objectifs et traitements précédents	2
Classification du problème	2
Première approche : modélisation à partir des métadonnées	3
KNN plus proches voisins #1	3
Réduction du nombre de classes	6
Les différentes approches explorées	8
Les modèles effectués avec un Bag of Word	8
Bag of Words avec n-grams et métadonnées	8
RandomForest #1 : Commentaires, titres, et métadonnées	9
RandomForest #2 : Commentaires et métadonnées	9
Bag of Words avec n-grams, sans métadonnées	10
XGBoost #1 : mots seuls	10
Algorithme du Bag of Words avec plusieurs n-gram	11
Les modèles réalisés avec un traitement avancé des commentaires	13
TF-IDF	13
TF-IDF avec RandomForest	13
TF-IDF avec XGBoost	14
Word-Embedding	15
KNN avec paramètre par défaut sur données standardisées:	15
RandomForest avec paramètre par défaut sur données standardisées:	16
RandomForest avec paramètre par défaut sur données non standardisées:	17
Test de Word-Embedding avec n-grams	17
Le modèle retenu	18
Evaluation du modèle	19
Perspectives d'amélioration	20

Rappel des objectifs et traitements précédents

L'objectif principal de notre étude est de développer un modèle prédictif performant pour évaluer la satisfaction des clients à partir de leurs commentaires. Nous avons effectué un scrapping de données sur le site Trustpilot, spécifiquement sur l'entreprise ShowRoom Privé. Ce procédé nous a permis de récupérer près de 170 000 avis.

Des variables ont été créées à partir des commentaires, relatives à divers aspects des commentaires tels que la taille, la ponctuation, la présence de majuscules. Les commentaires ont été nettoyés, en enlevant les mots sans significations (stopwords), la ponctuation, et en utilisant la lemmatisation pour normaliser les mots.

Classification du problème

Le problème d'interprétation des commentaires, et de déduction de l'avis laissé s'apparente à un cas d'analyse de sentiment. Selon les aspects que l'on cherche à traiter, le type de problème de machine learning diffère. Si on cherche à déterminer un avis entre 1 et 5, notamment en se basant sur les métadonnées des commentaires (taille du commentaire, nombre de majuscules, etc...) on pourrait prendre le problème comme un problème de régression. Si on cherche à déterminer, à partir des commentaires (et éventuellement des métadonnées), dans quelle "catégories" d'avis on est (catégorie 1, 2, 3, 4 ou 5), on est plutôt dans un problème de classification. On pourrait même aller dans d'autres idées, et tester si on pourrait déterminer les avis en fonction des dates à laquelle les avis ont été laissé, dans ces cas là nous serions dans un problème de série temporelle, mais intuitivement, on pense qu'il y a peu de chance d'obtenir des résultats satisfaisants si on prend ce prisme, aussi nous axerons notre étude sur la résolution, l'optimisation, d'un problème de classification.

Première approche : modélisation à partir des métadonnées

KNN plus proches voisins #1

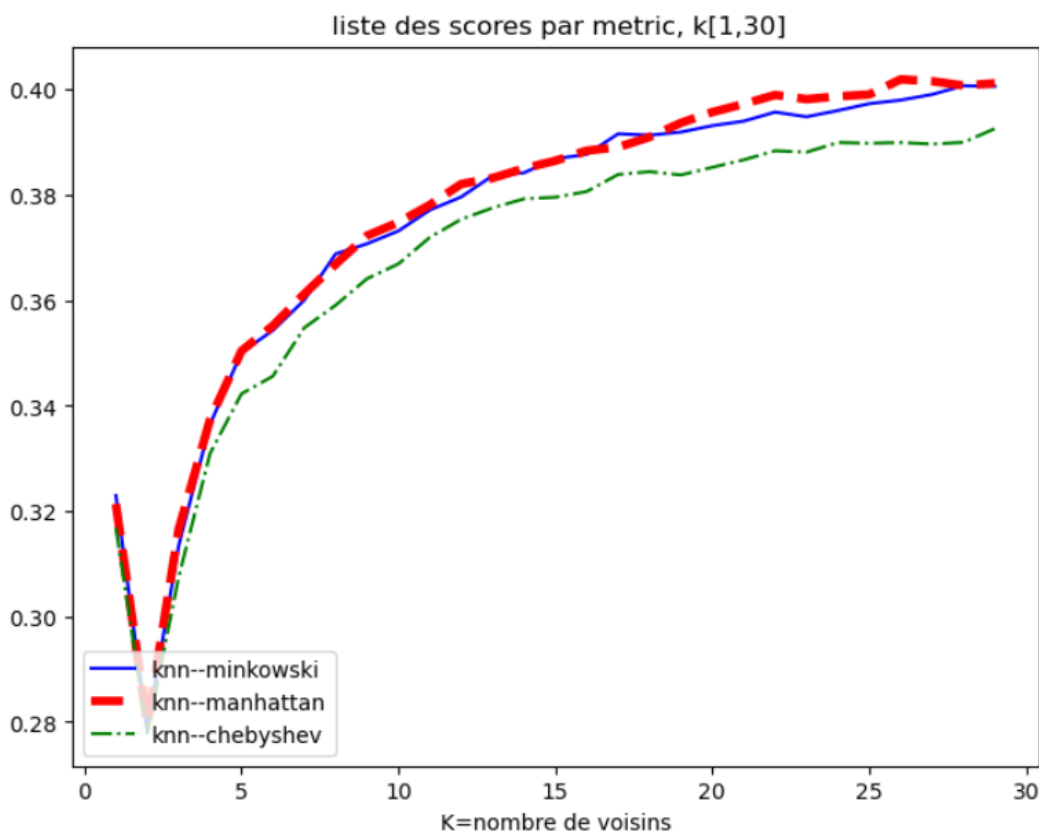
Nous nous sommes dans un premier temps intéressés au modèle Knn des plus proches voisins. Nous avons étudié, en premier lieu, les métadonnées de base de "data_preparee.csv" qui comportent, les longueurs du texte, nombre de signes de ponctuation, nombre de majuscules, nombre d'avis laissés par le client,

Choix des paramètres du modèles:

Nous avons effectué une recherche du nombre k de voisins qui nous donne le meilleur score. Aussi, nous avons utilisé les 3 métriques de mesures de distances entre les points, 'minkowski', 'manhattan' et 'chebyshev'.

Résultat :

Les scores obtenus avec les différentes métriques nous montrent que la métrique 'manhattan' est plus adaptée et donne un score plus élevé de la métrique 'chebyshev' et est similaire à celle de 'minkowski'. Toutefois, les scores obtenus ne sont pas du tout satisfaisants.



Sélection d'un modèle Knn avec k=10

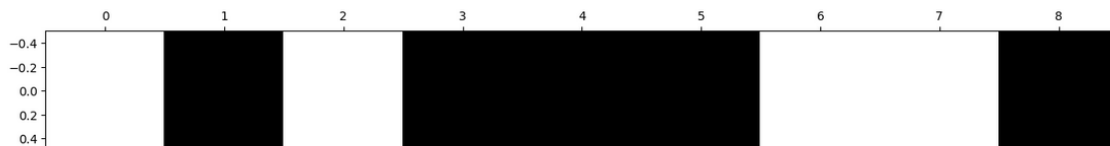
Nous avons retenu le modèle knn avec $k = 10$ avec la métrique 'manhattan'. Le rapport de classification donne des résultats pas très concluants.

	precision	recall	f1-score	support
1.0	0.32	0.34	0.33	4103
2.0	0.10	0.17	0.13	1771
3.0	0.30	0.28	0.29	6562
4.0	0.32	0.32	0.32	8923
5.0	0.55	0.51	0.53	12420
accuracy			0.37	33779
macro avg	0.32	0.32	0.32	33779
weighted avg	0.39	0.37	0.38	33779

Nous avons alors essayé de réduire la dimension du modèle afin de tester si cela permettait d'obtenir de meilleurs résultats.

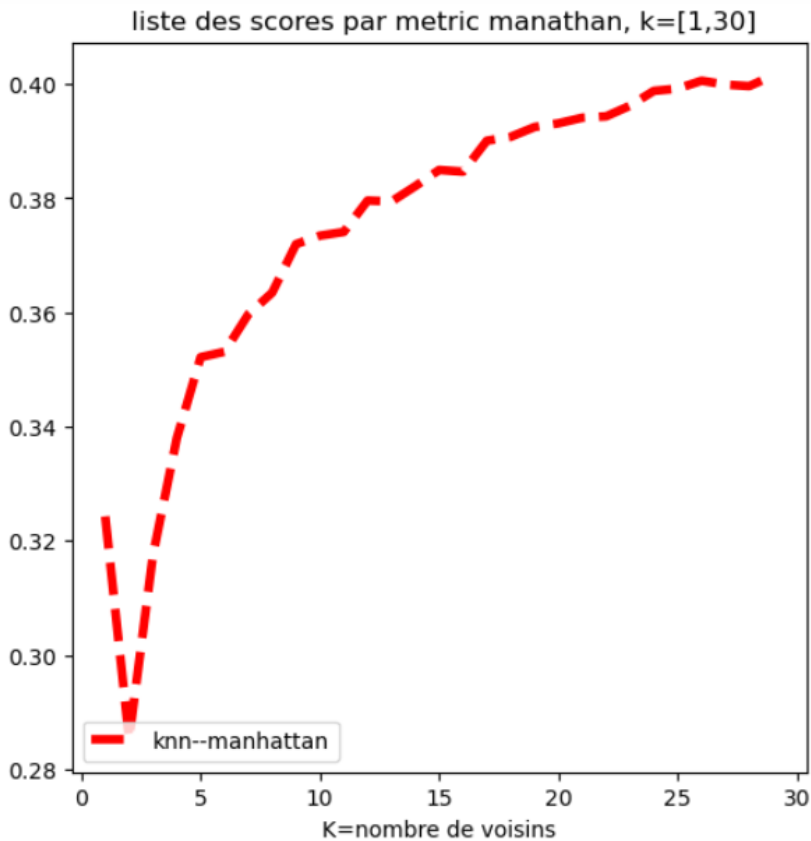
Réduction de dimensions :

Nous avons appliqué un preprocessing par filtration des features avec SelectKbest, nous avons gardé les variables apportant un maximum de variances et supprimé celles qui n'apportent peu ou pas d'informations. Nous nous sommes intéressés aux 50% des variables les plus importantes, soit 5 variables explicatives.



Modèle Knn avec les 5 meilleures variables :

Les scores ne sont pas meilleurs, avec un $k=10$ et 5 variables :



Ce premier modèle, bien que peu performant, nous a permis néanmoins de constater que les notes dans les extrêmes (1 et 5 étoiles) étaient un peu mieux identifiées que les notes au centre. Nous nous sommes interrogés sur la pertinence d'essayer de détecter la nuances des notes. Il est vrai qu'il n'est pas forcément aisé de déterminer spontanément si un commentaire positif a été associé à une note de 5 étoiles, ou alors de 4 étoiles. De l'autre côté du spectre, il n'est pas non plus simple de distinguer des nuances entre des avis négatifs qui ont reçu 3, 2 ou 1 étoile.

Réduction du nombre de classes

Choix de représentation du sentiment client :

Face au constat de notre premier modèle, nous avons fait le choix de chercher la meilleure manière de représenter la satisfaction clients à travers les notes et avis laissés sur le site. En effet, les notes que les clients attribuent pour exprimer si leurs expériences d'achats sont vécues comme positives ou négatives, voire, comme neutres, peuvent être exposées de différentes manières.

Voici un schéma que nous allons explorer pour avoir un point de repère.

Notes	1	2	3	4	5	
cas 1	1	2	3	4	5	5 classes
cas 2	Négatif	Négatif	Neutre	Positif	Positif	3 classes
cas 3	Négatif	Négatif	Négatif	Positif	Positif	2 classes
cas 4	Négatif	Négatif	Négatif	Négatif	Positif	2 classes
Le cas d'avis négatif ou positif est représenté par 0 et 1 respectivement Le cas d'avis négatif, neutre ou positif est représenté par -1, 0, +1 respectivement						

Nous avons testé ces différents cas au travers de l'étude des commentaires seuls, sans les métadonnées. Les données ont été préparées en appliquant un algorithme de bag of words, qui consiste à créer à partir des commentaires, une matrice avec un mot étudié par colonne. La variable prend 1 ou 0 en fonction de la présence, ou non, du mot dans le commentaire observé.

Nous avons calculé une représentation numérique de chaque mot utilisé dans les commentaires. Pour cette section nous nous limitons à un vocabulaire de taille = 4000 mots, les 4000 mots les plus utilisés.

```

X = df.Commentaire
y = df.star

X_train_text, X_test_text, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)

vectorizer = CountVectorizer(max_features=4000, stop_words = stopwords.words('french'))

X_train = vectorizer.fit_transform(X_train_text)
X_test = vectorizer.transform(X_test_text)

# taille du vocabulaire
voc = vectorizer.vocabulary_
print('Vocabulary size: ', len(voc))

```

Après calcul du bag of words, nous appliquons l'algorithme random Forest sur nos différents cas et calculons l'accuracy pour chaque situation.

Algorithme de RandomForest appliqué sur les # cas		Accuracy
Cas 1	les notes (1,2,3,4,5) représentées par (1,2,3,4,5)	0.55
Cas 2	les notes (1,2,3,4,5) représentées par (-1, -1, 0, 1, 1)	0.77
Cas 3	les notes (1,2,3,4,5) représentées par (0, 0, 0, 1, 1)	0.87
Cas 4	les notes (1,2,3,4,5) représentées par (0, 0, 0, 0, 1)	0.79

Le cas numéro 3 obtient le meilleur score, et le rapport de classification suivant :

le score en randomforest est : 0.87

	precision	recall	f1-score	support
0.0	0.84	0.83	0.83	13516
1.0	0.89	0.89	0.89	20263
accuracy			0.87	33779
macro avg	0.86	0.86	0.86	33779
weighted avg	0.87	0.87	0.87	33779

Classe prédite	0.0	1.0
Classe réelle		
0.0	11181	2335
1.0	2220	18043

Résultat :

D'après les différents accuracy obtenues, nous remarquons que la représentation du sentiment général du commentaire client est plutôt bien géré par l'algorithme dans le cas où positif = 1 pour les notes (4* et 5*) et le cas négatif = 0 pour les notes (1*, 2*, 3*).

Aussi, nous avons la meilleure précision dans l'attribution des classes et en même temps un équilibre assez correct dans la répartition entre les deux classes 0 et 1.

Dans la suite de notre projet, nous retenons ce cas pour les différentes situations à étudier.

Les différentes approches explorées

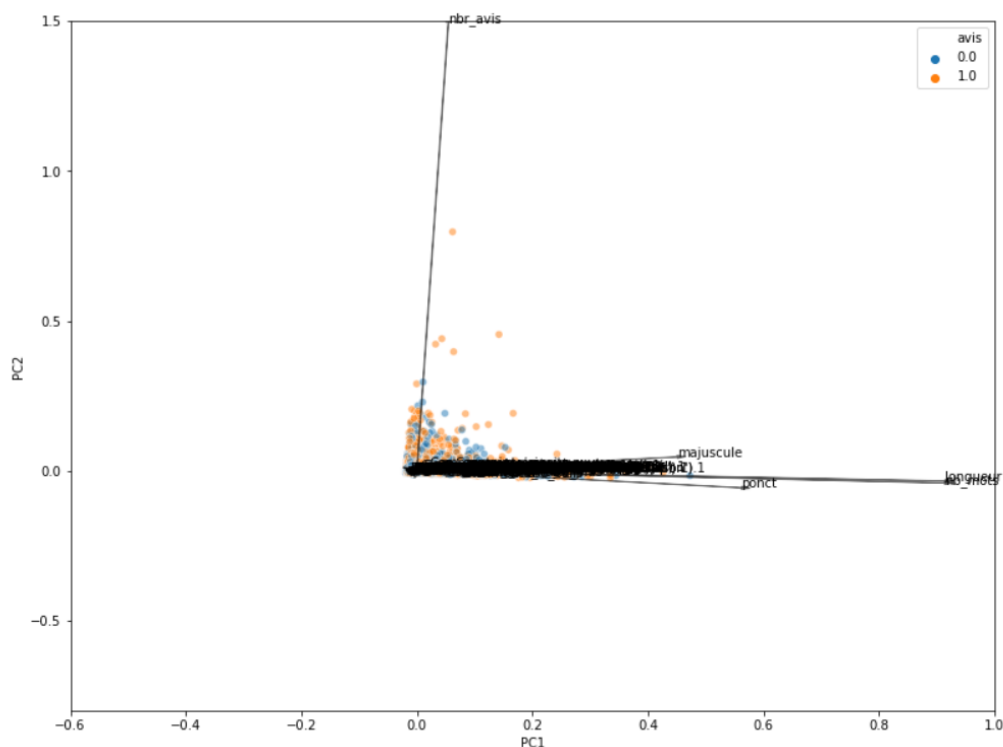
Les modèles effectués avec un Bag of Word

Bag of Words avec n-grams et métadonnées

Une première vague de modèles a été testée à partir des données transformées complètes (les métadonnées, les commentaires, les titres, et les n-grams découlant des commentaires et des titres). Ne connaissant pas dans un premier temps la méthode CountVectorizer (qui permet de créer une matrice avec pour chaque colonne, un des mots contenus dans les commentaires, et pour chaque ligne associée, un 0 ou un 1 selon la présence du mot), nous avons réalisé ce travail de dichotomisation par le biais d'une boucle, qui testé la présence, dans chaque commentaire, des 20 mots les plus utilisés, les 20 2-grams les plus utilisés, les 20 3-grams les plus utilisés. La même chose a été réalisée sur les titres. Les données ont été standardisées afin de réduire l'écart-type entre les différentes variables.

Nous avons ensuite procédé à une réduction de dimension pour ne conserver que les variables permettant de conserver 90% de la variance expliquée, à l'aide d'une Analyse en Composante Principale. 35 variables ont été retenues.

Si on regarde dans le détail les variables qui semblent avoir un poids important dans l'analyse, on retrouve en premier plan, sans trop de surprises, les métadonnées :



RandomForest #1 : Commentaires, titres, et métadonnées

Un premier modèle de RandomForest a été testé, renvoyant un score global de 81%. Dans le détail, les avis positifs arrivaient à être mieux détectés que les avis négatifs.

	precision	recall	f1-score	support
0	0.76	0.77	0.76	13717
1	0.84	0.83	0.84	20062
accuracy			0.81	33779
macro avg	0.80	0.80	0.80	33779
weighted avg	0.81	0.81	0.81	33779

RandomForest #2 : Commentaires et métadonnées

Un autre test avec un modèle de RandomForest a été réalisé, mais cette fois sans les titres des commentaires (ainsi que les métadonnées associées), car l'objectif final est de ne se servir que des commentaires, sans les titres.

L'algorithme de RandomForest fait un meilleur score, mais pas si éloigné du précédent, avec une précision de 83%. Même observation que précédemment, le modèle détecte mieux les avis positifs.

	precision	recall	f1-score	support
0	0.79	0.79	0.79	13496
1	0.86	0.86	0.86	20283
accuracy			0.83	33779
macro avg	0.82	0.82	0.82	33779
weighted avg	0.83	0.83	0.83	33779

Bag of Words avec n-grams, sans métadonnées

Dans cette section, nous allons former un modèle de classification à partir des données de base. Nous allons nous intéresser uniquement aux commentaires (déjà traités et nettoyés) et à leurs notes associées.

Nous utiliserons la technique du bag of words avec différentes fonctionnalités (plusieurs de choix de n-gram) avec l'application de deux algorithmes d'apprentissages, RandomForest et XGBoost, et comparer leurs résultats respectifs.

Le bag of words permet de créer facilement une matrice prenant 0 ou 1 en fonction de la présence, ou non, d'un mot ou n-gram.

XGBoost #1 : mots seuls

Différents tests ont été réalisés, avec différentes tailles de jeux de données (car la base entière prenait trop de temps à calculer avec l'algorithme).

Après plusieurs itération, nous sommes arrivé à utiliser 50% de notre jeu de données, avec un learning rate abaissé à 0.5, permettant, avec un temps d'entraînement contraint d'obtenir un modèle, uniquement basé sur les commentaires, avec un score de 84%, mais avec un recall néanmoins moindre que le modèle de RandomForest avec les métadonnées pour les avis négatifs.

	precision	recall	f1-score	support
0	0.85	0.74	0.79	20276
1	0.84	0.91	0.87	30392
accuracy			0.84	50668
macro avg	0.84	0.83	0.83	50668
weighted avg	0.84	0.84	0.84	50668

Algorithme du Bag of Words avec plusieurs n-gram

Nous avons étudié différents cas de figures dans l'application de notre bag of words, notamment autour du traitement des n-grams avec les configurations suivantes du BoW :

- ngram_range = 1, n-gram = 2 et n-gram = 3
- ngram_range = 1 à 2, (1 et 2 réunis)
- ngram_range = 1, 2 et 3 (1, 2, 3 réunis)

ngram_range = 1 correspond aux mots seuls

Nous avons testé les résultats au travers de deux algorithmes :

- RandomForest
- XGBoost

Dans le tableau suivant, les accuracy obtenus dans chaque configuration :

Ngram du BoW	Accuracy RandomForest	Accuracy XGBoost
ngram = 1	0.87	0.83
ngram = 2	0.82	0.78
ngram = 3	0.42	0.78
ngram = 1+2	0.87	0.86
ngram = 1 + 2 + 3	0.87	0.86
ngram = 1 + 2 + 3 + 4	0.87	0.86

d'après ce tableau les valeurs qui nous intéressent le plus sont à priori issu de deux configurations :

- ngram = 1
- ngram = 1 + 2 cumulés

Il est utile de noter que dans le cas où les ngrams = 1+2+3 et ngrams = 1+2+3+4 donnent des résultats similaires aux précédents. Donc nous allons nous limiter aux configuration retenus du fait que nous n'avons pas plus d'informations et de précisions supplémentaires.

Voici le rapport détaillé de la classification des cas retenus

Ngram BoW	du	RandomForest				
ngram = 1		le score en randomforest est : 0.87				
		precision	recall	f1-score	support	
	0.0	0.84	0.83	0.83	13516	
	1.0	0.89	0.89	0.89	20263	
	accuracy			0.87	33779	
	macro avg	0.86	0.86	0.86	33779	
	weighted avg	0.87	0.87	0.87	33779	
ngram = 1+2		le score en randomforest est : 0.87				
		precision	recall	f1-score	support	
	0.0	0.83	0.83	0.83	13516	
	1.0	0.89	0.89	0.89	20263	
	accuracy			0.87	33779	
	macro avg	0.86	0.86	0.86	33779	
	weighted avg	0.87	0.87	0.87	33779	
Ngram BoW	du	XGBoost				
ngram = 1		le score de xgboost en unigram est : 0.83				
		precision	recall	f1-score	support	
	0.0	0.85	0.70	0.77	13516	
	1.0	0.82	0.92	0.87	20263	
	accuracy			0.83	33779	
	macro avg	0.84	0.81	0.82	33779	
	weighted avg	0.83	0.83	0.83	33779	
ngram = 1+2		le score de xgboost classifieur est : 0.86				
		precision	recall	f1-score	support	
	0.0	0.87	0.78	0.82	13516	
	1.0	0.86	0.92	0.89	20263	
	accuracy			0.86	33779	
	macro avg	0.86	0.85	0.86	33779	
	weighted avg	0.86	0.86	0.86	33779	

Les modèles réalisés avec un traitement avancé des commentaires

Conscient que nos précédents modèles ne faisaient que compter et vérifier la présence de certains de certains mots (ou n-grams) dans les commentaires, sans vraiment chercher à caractériser et comprendre l'enchaînement des mots dans les phrases, ou à pondérer les variables selon la rareté des mots, nous avons dans un second temps travaillé sur différents traitements des commentaires, notamment via des procédés de TF-IDF et vectorisation de commentaires, afin de donner plus ou moins d'importance aux mots selon leurs emplacements dans les phrases, selon les mots qui précèdent et succèdent chaque mots.

TF-IDF

TF-IDF avec RandomForest

Le TF-IDF (term frequency-inverse document frequency), évolue les mots et n-grams, en affectant un poids qui augmente proportionnellement au nombre d'occurrences dans l'ensemble des commentaires, tout en cherchant à réduire l'impact des termes trop fréquents et peu discriminant dans la segmentation des commentaires.

Une fois réalisé le retraitement de la base via le procédé de TF-IDF sur les n-grams 2 à 4, nous avons tester un modèle de Random Forest sur la matrice obtenue, renvoyant ici un score de 83% et des métriques similaires aux précédents tests :

	precision	recall	f1-score	support
0	0.81	0.73	0.77	20276
1	0.83	0.89	0.86	30392
accuracy			0.83	50668
macro avg	0.82	0.81	0.82	50668
weighted avg	0.82	0.83	0.82	50668

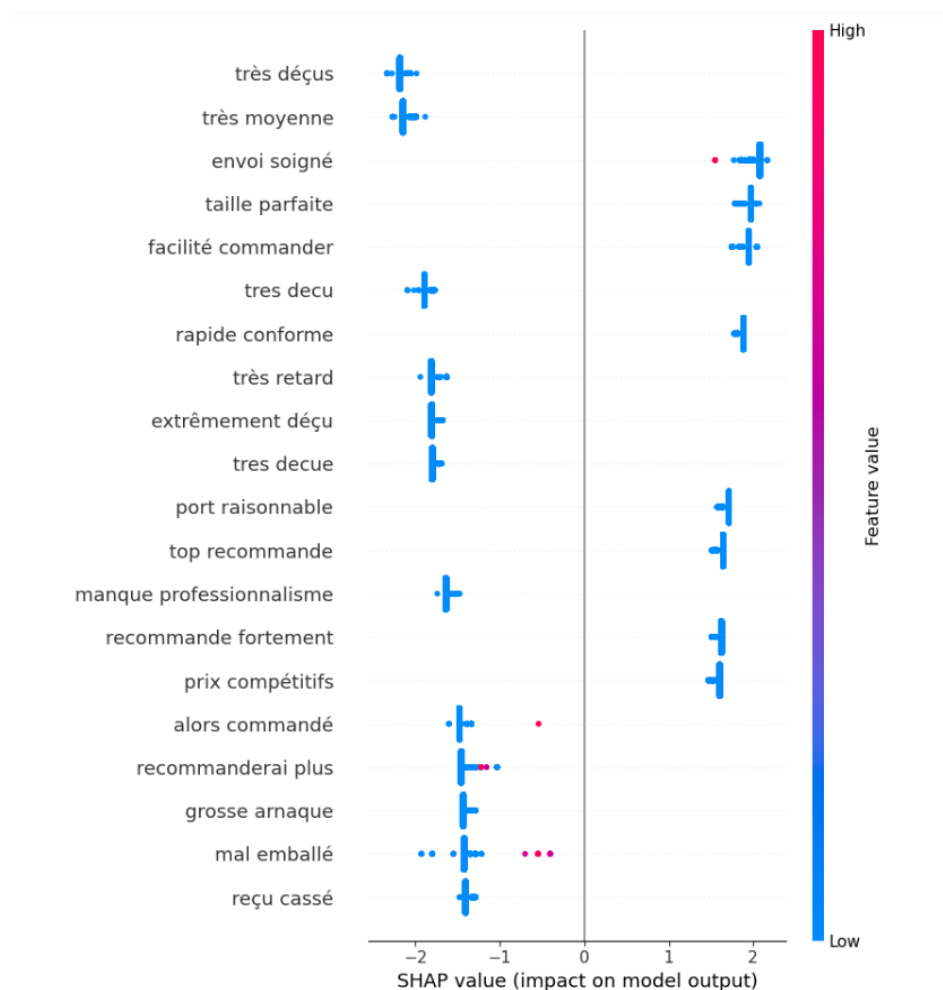
Le traitement par TF-IDF est ici assez peu concluant, ou en tous les cas n'apporte pas de précision supplémentaire.

TF-IDF avec XGBoost

Le même procédé de retraitement via un procédé de TF-IDF a été réalisé, suivi d'une classification via XGBoost. Les paramètres du modèle XGBoost ont été affinés à l'aide d'un GridSearchCV. Le même modèle en sortie donnait les résultats suivants :

	precision	recall	f1-score	support
0	0.84	0.70	0.77	20276
1	0.82	0.91	0.86	30392
accuracy			0.83	50668
macro avg	0.83	0.81	0.82	50668
weighted avg	0.83	0.83	0.83	50668

Cela n'apporte pas une différence marquante par rapport aux précédents modèles. A l'aide d'une analyse SHAP, nous pouvons identifier quels n-grams ont le plus d'importance dans ce modèle :



Word-Embedding

Démarche :

Nous avons essayé de retraiter les commentaires avec une méthode de word-embedding, qui caractérise chacun des mots par de multiples vecteurs, dont la valeur dépend de la proximité et la récurrence des autres mots autour des mots caractérisés. Les mots sont ainsi transformés en de multiples variables, que l'on peut retraiter par la suite par des modèles de classification.

Suite à l'entraînement d'un modèle de word Embedding (Word2Vec), caractérisant et vectorisant les mots (utilisés plus de 50 fois dans la base) dans un espace à 4 000 dimensions mots les plus fréquents des commentaires, la matrice de sortie a été réduite par une ACP gardant 90% de la variance expliquée. Cette étape était nécessaire, car les données étaient très volumineuses par rapport à notre puissance de calcul (données supérieures au 25Go limite de RAM de notre environnement).

Dans cette section nous avons appliqué plusieurs modèles avec recherche des meilleurs paramètres afin d'optimiser les scores obtenus et cherché un modèle stable et qui s'ajuste aussi bien sur les échantillons train et test de nos données.

En premier lieu, nous avons comparé l'algorithme randomforest et knn plus proche voisin sur les données de base versus données standardisées. Ensuite, afin d'affiner notre modèle et tenter d'optimiser les sorties, avons appliqué un XGBoost avec une recherche croisée à l'aide d'un grid search. Aussi, nous avons recherché les meilleurs paramètres pour l'algorithme knn et randomforest

Section de standardisation des données:

Dans cette partie, nous allons essayer d'appliquer un standard scaler sur nos données en sortie après réduction de dimension. L'intérêt est de savoir si une standardisation apporterait quelques en plus en performance que lorsqu'on traite directement dans les vecteurs après la vectorisation suite à l'application du word2vec.

KNN avec paramètre par défaut sur données standardisées:

Le Knn donne en premier essai des résultats satisfaisants, les scores entre les échantillons train et test sont assez proches et aussi la mean_squared_error sont de petites valeurs et rapprochées.

Notre modèle s'ajuste correctement sur les deux ensembles. Nous allons essayer d'approfondir les paramètres pour une meilleure performance si possible.

	precision	recall	f1-score	support
0	0.79	0.84	0.81	16901
1	0.89	0.85	0.87	25322
accuracy			0.85	42223
macro avg	0.84	0.85	0.84	42223
weighted avg	0.85	0.85	0.85	42223

Classe prédite 0 1  

Classe réelle

0	14159	2742
1	3732	21590



le score sur l'échantillon train est : 0.88
le score sur l'échantillon test est : 0.85

mean_squared_error train : 0.12
mean_squared_error test : 0.15

RandomForest avec paramètre par défaut sur données standardisées:

Les résultats pour ce modèle avec paramètres par défaut montrent un surapprentissage sur l'échantillon train. Bien que les scores ne soient pas mauvais sur l'échantillon test, nous allons tenter par la suite d'affiner les scores.

	precision	recall	f1-score	support
0	0.84	0.78	0.81	16901
1	0.86	0.90	0.88	25322
accuracy			0.85	42223
macro avg	0.85	0.84	0.84	42223
weighted avg	0.85	0.85	0.85	42223

Classe prédite 0 1  

Classe réelle

0	13125	3776
1	2574	22748

le score sur l'échantillon train est : 0.99
le score sur l'échantillon test est : 0.85

mean_squared_error train : 0.16
mean_squared_error test : 0.17

RandomForest avec paramètre par défaut sur données non standardisées:

À première vue, l'algorithme répond assez correctement dans le cas des données non standardisées. les scores et la mse sont assez bons sur le train et test, ce qui indique que le modèle ne fait pas de d'over ni d'underfitting.

	precision	recall	f1-score	support
0	0.83	0.74	0.78	16901
1	0.84	0.90	0.87	25322
accuracy			0.84	42223
macro avg	0.83	0.82	0.83	42223
weighted avg	0.83	0.84	0.83	42223

Classe prédite	0	1
Classe réelle		
0	12569	4332
1	2622	22700

```
print("le score sur l'échantillon train est :", round(rf.
print("le score sur l'échantillon test est :", round(rf.
```

```
le score sur l'échantillon train est : 0.85
```

```
le score sur l'échantillon test est : 0.84
```

```
y_pred_train = rf.predict(X_train)
y_pred_test = rf.predict(X_test)
```

```
print(' mean_squared_error train :', round(mean_squared_e
print(' mean_squared_error test :', round(mean_squared_e
```

```
mean_squared_error train : 0.15
```

```
mean_squared_error test : 0.16
```

Test de Word-Embedding avec n-grams

Nous avons entraîné un modèle de word-embedding, incluant cette fois si l'étude de n-grams (bi-grams et tri-grams). Nous avons ajusté un modèle d'XGBoost à l'aide d'un grid-search. Les performances affichées ne semblent pas meilleures qu'un modèle de word-embedding simple, aussi, nous n'avons pas retenu ce modèle.

Le modèle retenu

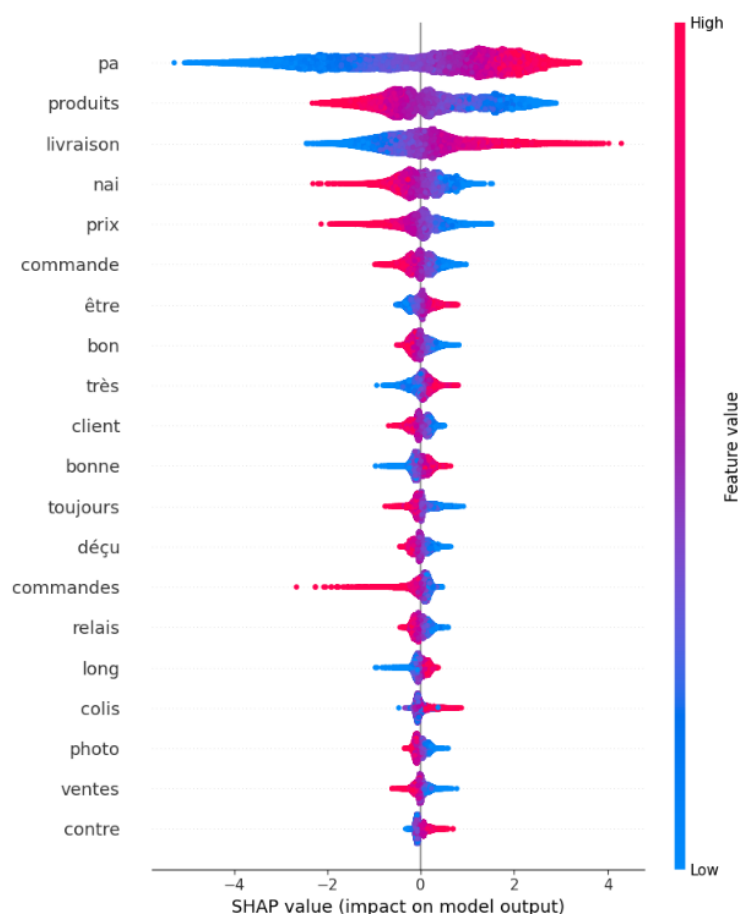
Après plusieurs ajustements, un dernier test a été réalisé avec un extrême gradient boosting, donc les hyper-paramètres ont été ajustés avec un Grid Search. Plusieurs tests ont été réalisés autour de réglages du learning-rate [1, 0.5, 0.3, 0.25, 0.225, 0.2, 0.175, 0.15, 0.1, 0.05], de la profondeur maximale [4, 5, 6, 7, 8] et du nombre d'estimateurs [80, 100, 120, 140, 150, 160, 180, 200]. La meilleure combinaison fut un learning rate de 0,225, une profondeur maximale de 5 et 150 estimateurs, donnant les résultats suivants :

	precision	recall	f1-score	support
0	0.84	0.84	0.84	16901
1	0.89	0.90	0.89	25322
accuracy			0.87	42223
macro avg	0.87	0.87	0.87	42223
weighted avg	0.87	0.87	0.87	42223

Le score d'accuracy de l'ensemble d'entraînement est de 0.8926, ce qui est assez proche du score d'accuracy de l'ensemble de test. Il ne semble pas que nous soyons face à un cas d'overfitting, ce qui est très bien.

Représentation des facteurs les plus impactants du modèle via SHAP :

Si on étudie les facteurs (mots) ayant le plus d'impact dans le modèle, on constate que la négation "pas" est le mot ayant le plus d'impact dans le modèle, influant plutôt vers la négative l'évaluation d'un commentaire.



Evaluation du modèle

Le modèle, sur le papier, arrive assez bien à distinguer les commentaires positifs des commentaires négatifs, malgré la relative simplicité de ce dernier.

Plusieurs tests ont été réalisés :

Commentaire: service client déplorable Sentiment prédit: Negative	Commentaire: Je suis content Sentiment prédit: Positive
Commentaire: très déçu Sentiment prédit: Negative	Commentaire: Pas OK Sentiment prédit: Positive
Commentaire: livraison en retard, très mécontent Sentiment prédit: Negative	Commentaire: pas bon Sentiment prédit: Negative
Commentaire: très content Sentiment prédit: Positive	Commentaire: super lent, très en retard, livraison catastrophique Sentiment prédit: Negative
Commentaire: livraison en retard, très content Sentiment prédit: Positive	Commentaire: pas lent, pas en retard Sentiment prédit: Negative
Commentaire: livraison très en avance Sentiment prédit: Positive	Commentaire: Service client réactif Sentiment prédit: Positive
Commentaire: Je ne suis pas content! Sentiment prédit: Negative	Commentaire: RAS Sentiment prédit: Positive
Commentaire: J'ai commandé en début de mois, c'est arrivé 4 semaines plus tard. Je n'ai jamais vu un service aussi lent !!! Le SAV ne répond pas quand je les appelle. A fuir !!! Sentiment prédit: Negative	Commentaire: C'est la dernière fois que je commande chez eux Sentiment prédit: Positive

Bilan : A première vue, le modèle semble plutôt bien classer les commentaires, hors certaines formes de phrase, tournées à la négative, mais assez peu spontanées : "Pas OK", "pas lent, pas en retard". Ou des phrases avec des sentiments contradictoires : "livraison en retard, très content". Le modèle perçoit assez mal l'ironie. On a aussi des phrases mal classées, comme "C'est la dernière fois que je commande chez eux", dont on a du mal à identifier les causes, mais cela fait partie de 13% d'imprécision du modèle.

Perspectives d'amélioration

Il existe plusieurs pistes qui pourraient permettre de donner de meilleurs résultats :

-Tester un modèle de Word-Embedding avec une moindre réduction de dimension, ce qui nécessite, soit d'augmenter la RAM de nos systèmes, soit de trouver une technique permettant d'éviter que l'intégralité des données soient présentes en même temps dans la RAM.

-Tester d'autres algorithmes de classification, notamment essayer de la classification avec des réseaux de neurones.

-Tout en restant sur un XGBoost, on pourrait peut-être également obtenir de meilleurs scores en ajustant d'autres hyperparamètres.

-Nous aurions également eu une bien meilleure compréhension de la langue si nous nous étions basés sur des Large Language Model déjà pré-entraînés en open source, comme le modèle Llama (1 ou 2) de Meta. C'est sans doute le scénario le plus efficace sur notre problématique, mais nous aurions perdu l'intérêt pédagogique d'entraîner un modèle.

-Nous aurions pu explorer les techniques de POS Tagging dans la préparation de données, afin de faire un meilleur tri des mots à conserver au moment de la préparation des données.

-Il aurait été satisfaisant de réussir à revenir sur modèle d'évaluation en nombre d'étoiles, comme initialement pensé. Un dernier essai a été réalisé à partir de notre modèle entraîné de word-embedding et un XGBoost, dont les hyperparamètres ont été ajustés à l'aide d'un grid search. Les résultats donnent la même conclusion que précédemment : la détection de notes intermédiaires (2, 3 ou 4 étoiles) est assez compliquée :

	precision	recall	f1-score	support
0	0.54	0.65	0.59	5411
1	0.31	0.11	0.16	3710
2	0.46	0.51	0.48	7780
3	0.52	0.42	0.46	10986
4	0.67	0.79	0.73	14336
accuracy			0.56	42223
macro avg	0.50	0.49	0.48	42223
weighted avg	0.54	0.56	0.55	42223