

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

Rapport final

Cohorte de novembre 2022 - parcours Data Scientist

Adnane MOUZAOU

François ROUXELIN

Sous la supervision de Yohan COHEN



DataScientest

DataScientest.com

Agrément organisme de formation 11755665975

09 80 80 79 49

2 place de Barcelone, 75016 Paris

Sommaire

Introduction	2
Objectifs	3
Cadre	3
Webscraping et données personnelles	4
Disponibilité des données	4
Utilisation principale des données	5
Web scraping des avis clients	5
Pertinence	6
Pre-processing et feature engineering	7
Exploration du dataset issu du scraping de ShowRoomPrivé	7
Analyse et exploration rapide du dataframe	8
Choix de l'intervalle de dates	8
Retraitement des commentaires	9
Visualisations et Statistiques	10
Répartition de notre variable cible, le nombre d'étoiles	10
Répartition et distribution du nombre d'avis postés	11
Etude de la répartition pour l'année 2015 et 2016	13
Relation entre "star" et les caractéristiques des commentaires	16
Analyse des commentaires	19
Analyse des titres	22
Classification du problème	25
Première approche : modélisation à partir des métadonnées	26
KNN plus proches voisins #1	26
Réduction du nombre de classes	29
Les différentes approches explorées	31
Les modèles effectués avec un Bag of Word	31
Bag of Words avec n-grams et métadonnées	31
RandomForest #1 : Commentaires, titres, et métadonnées	32
RandomForest #2 : Commentaires et métadonnées	32
Bag of Words avec n-grams, sans métadonnées	33
XGBoost #1 : mots seuls	33
Algorithme du Bag of Words avec plusieurs n-gram	34
Les modèles réalisés avec un traitement avancé des commentaires	36
TF-IDF	36
TF-IDF avec RandomForest	36
TF-IDF avec XGBoost	37
Word-Embedding	38
KNN avec paramètre par défaut sur données standardisées:	38
RandomForest avec paramètre par défaut sur données standardisées:	39
RandomForest avec paramètre par défaut sur données non standardisées:	40
Test de Word-Embedding avec n-grams	40
Le modèle retenu	41
Evaluation du modèle	42
Perspectives d'amélioration	43
Difficultés rencontrées lors du projet	44
Bilan	45
Références	46

Introduction

La satisfaction client est un élément essentiel pour toute entreprise souhaitant conserver sa clientèle et améliorer sa réputation. Dans ce contexte, l'analyse des commentaires et des avis des clients est devenue une pratique courante pour évaluer la qualité de la supply chain, la conformité des produits/services aux attentes du marché et pour identifier les points d'amélioration nécessaires.

Cependant, l'analyse manuelle de ces commentaires peut être fastidieuse et chronophage. Dans ce projet, nous proposons une approche automatisée pour extraire de l'information de commentaires afin de prédire la satisfaction d'un client. Plus précisément, nous visons à prédire le nombre d'étoiles associé à chaque commentaire à partir de données disponibles sur Trusted Shops et Trustpilot.

Cette étude présente un intérêt à la fois économique, technique et scientifique. Sur le plan économique, les résultats de cette étude peuvent aider les entreprises à mieux comprendre les attentes de leurs clients et à améliorer leur satisfaction. Ils peuvent également contribuer à la réduction des coûts liés à l'analyse manuelle des commentaires. D'un point de vue technique, l'utilisation de techniques de traitement du langage naturel et de machine learning permet de transformer des données non structurées en informations exploitables. Enfin, d'un point de vue scientifique, cette étude permet d'explorer les limites de la prédiction de la satisfaction client à partir de commentaires textuels.

En résumé, l'objectif de ce projet est de proposer une approche automatisée pour prédire la satisfaction client à partir de commentaires textuels, avec des implications importantes pour les entreprises, la technologie et la recherche scientifique.

Objectifs

L'objectif principal de notre étude est de développer un modèle prédictif performant pour évaluer la satisfaction des clients à partir de leurs commentaires. Pour cela, nous avons eu accès à des données de deux sources différentes : les avis vérifiés de Trusted Shops et les avis d'internautes de Trustpilot. Nous avons également enrichi notre base de données grâce au web scraping, en récupérant des données associées à l'entreprise Showroom Privé.

Dans le cadre de notre analyse, nous chercherons à comprendre les variables qui peuvent avoir un impact sur la satisfaction client, en examinant divers aspects des commentaires tels que la taille, la ponctuation, la présence de majuscules, etc. Nous effectuerons également une étape de nettoyage des données, en enlevant les mots sans signification ou redondants tels que les stopwords, et en utilisant la lemmatisation pour normaliser les mots.

En somme, nous chercherons à comprendre les facteurs clés qui contribuent à la satisfaction des clients et à développer un modèle prédictif robuste pour prédire leur note directement à partir de leurs commentaires. Cette étude présente des enjeux économiques, techniques et scientifiques importants, en permettant notamment aux entreprises de mieux comprendre les besoins de leurs clients et d'adapter leur offre en conséquence.

Cadre

Dans le cadre de notre projet, nous avons utilisé deux jeux de données pour atteindre nos objectifs. Le premier jeu de données a été fourni par notre organisme de formation et contenait des informations sur les entreprises ShowRoom et VeePee, récupérées à partir des données des sites TrustedShop et Trustpilot. Ce jeu de données comprenait environ 20 000 lignes.

Ce jeu de données a été enrichi en réalisant un scrapping de données sur le site Trustpilot, spécifiquement sur l'entreprise ShowRoom Privé. Ce procédé nous a permis d'ajouter près de 170 000 avis supplémentaires, et également de récupérer deux nouvelles variables, qui est le nombre d'avis laissés par un utilisateur, que nous avons nommé 'nb_avis', ainsi que le titre de l'avis, que nous avons nommé "Titre". Nous disposons ainsi d'une base de données conséquente pour entraîner notre modèle. Afin de limiter le nombre de doublons, nous avons choisi de ne conserver que la base de données scrappées.

Webscraping et données personnelles

Disponibilité des données

Les données ayant fait l'objet du scraping sont disponibles publiquement sur le site de trustpilot. Concernant le volet réglementation, de manière générale, le scraping de données est un domaine juridique complexe et il est important de prendre en compte les lois et les règles applicables à chaque site web et à chaque pays.

Le règlement général sur la protection des données, ou RGPD, s'applique aux données personnelles. Il s'agit de toute information personnelle identifiable (IPI) qui pourrait être utilisée pour identifier directement ou indirectement une personne physique.

Pour des personnes physiques, voici un aperçu de ce que relate **CNIL** (France) :

- nom, prénom, pseudonyme, date de naissance;
- photos, enregistrements sonores de voix;
- numéro de téléphone fixe ou portable, adresse postale, adresse email;
- adresse IP, identifiant de connexion informatique ou identifiant de cookie;
- empreinte digitale, empreinte rétinienne, etc..
- numéro de plaque d'immatriculation, de sécurité sociale ou de pièce d'identité.

L'identification n'est pas toujours possible à partir d'une seule de ces données personnelles mais peut être réalisée par un croisement de ces dernières.

Dans notre cas, du web scraping, une partie des informations récoltées contiennent le nom et prénom des clients et autres indications (liste ci-dessus). Il est utile et nécessaire de préciser qu'il ne nous est pas possible d'identifier les clients d'une manière directe ou indirecte avec ces seules informations, ni avec complément et croisement avec d'autres informations collectées. Aussi, ces données sont :

- Accessibles et publics sur le web
- Stockées de manière sécurisée et conformément aux meilleures pratiques.
- Ne sont pas vendues ou partagées avec des tiers.
- Non utilisées à des fins de démarchages commerciales
- Ne subissent pas de traitement à des fins de prospection.

Utilisation principale des données

L'usage principal des informations scrappées est l'étude statistique et scientifique des commentaires publiés par le client. L'objectif est de faire une analyse de sentiment sur les textes publiés par les clients exprimant leurs retours d'expérience suite à un achat.

Les données qui peuvent être sensibles (nom et prénom) ne seront pas étudiées en tant que telles, aucun lien ne sera fait avec des personnes physiques ou morales. Notre analyse portera principalement sur une étude qualitative et quantitative des avis (textes) publiés.

Web scraping des avis clients

Dans le jeu de données initial, nous avons constaté que de nombreuses colonnes et lignes contiennent de grandes quantités de valeurs manquantes. Nous avons ainsi procédé à un webscraping pour enrichir la base de données.

Pour ce faire, nous avons choisi le site Trustpilot pour extraire les avis postés pour les entreprises VeePee et ShowRoomPrivée.

Site support : trustpilot
Entreprises recherchées : VeePee et ShowRoomPrivé
Librairie utilisée : Sélénium/Python

The screenshot shows a Trustpilot review for Showroomprive.com. The review is from a user 'S.' with 1 review and a location of FR. The review has a 5-star rating and is marked as 'Vérifié' (Verified). The review text is 'Tout est conforme à la commande. Aucune surprise lors du déballage et de la réception du colis.' The date of the experience is '15 mars 2023'. The review is titled 'Tout est conforme à la commande'. The review is dated 'Il y a 2 jours'. The review is titled 'Réponse : Showroomprive.com' and the response text is 'Bonjour Francis, Vous nous faites part de votre satisfaction concernant la conformité de votre commande ainsi que son délai de livraison rapide.'

Annotations for data extraction:

- Client Information:**
 - Nom du client
 - Nombre total d'avis postés
 - Pays
- Review Information:**
 - Note attribuée
 - Date de publication de l'avis
 - Titre du commentaire
 - Texte du commentaire
- Transaction Information:**
 - Date d'achat d'article

Les avis sont affichés sur le site selon le format ci-dessus, nous avons extrait les informations suivantes :

- nom_client : Nom du client,
- note_avis : Note attribuée à l'achat
- date_avis : Date de publication de l'avis
- date_achat : Date d'achat
- nbr_avis : Nombre total des avis postés par le même client
- titre_avis : Titre du commentaire
- texte_avis : Texte du commentaire
- pays : Pays

Résultat du scraping

	ShowRoomPrivé	VeePee
Nombres d'avis récoltés	166 897	4074

Pertinence

Le jeu de données fourni de base comporte plusieurs variables dont le nombre d'étoiles ('star'), notre variable cible, et le commentaire laissé par l'internaute ('Commentaire'), commentaire qu'il conviendra d'étudier pour en déduire la note laissée par l'internaute. D'autres indicateurs, comme la date de publication de l'avis ('date'), le délai entre la date de commande et la date de l'avis ('ecart') ainsi que le nombre d'avis laissés par l'utilisateur ('nb_avis') pourraient nous aider à prédire à la note. Les autres indicateurs caractérisant les commentaires, qui seront créés par la suite, auront une importance primordiale dans les premiers modèles de prédictions que nous développerons. Notre jeu de données étant constitué majoritairement de données issues de ShowRoomPrivé, il pourra être intéressant de tester dans un temps futur la fiabilité du modèle proposé avec les données d'autres entreprises et services.

Pre-processing et feature engineering

Exploration du dataset issu du scraping de ShowRoomPrivé

Ce dataset comporte 168 897 avis récoltés.

il contient 8 colonnes :

- **nom_client** : Nom du client,
- **note_avis** : Note attribuée à l'achat
- **date_avis** : Date de publication de l'avis
- **date_achat** : Date d'achat
- **nbr_avis** : Nombre total des avis postés par le même client
- **titre_avis** : Titre du commentaire
- **texte_avis** : Texte du commentaire
- **pays** : Pays

nom_client	note_avis	date_achat	date_avis	nbr_avis	texte_avis	pays	titre_avis
Agani	1.0	: 26 août 2022	2023-02-27	2.0	Je n'ai jamais reçu ma commande. J'ai écrits à ...	FR	Je n'ai jamais reçu ma commande
CHANTAL SLATKINE	1.0	: 26 février 2023	2023-02-27	2.0	J'ai commandé 2 colliers. L'un est OK l'autre ...	FR	J'ai commandé 2 colliers
LDC	1.0	: 26 février 2023	2023-02-27	2.0	J ai commandé des airpods reconditionnés, dit ...	FR	Très déçue de la dernière commande.
anass jeffal	1.0	: 27 octobre 2022	2023-02-27	1.0	Produit acheté en Septembre 2022 retourné le m...	FR	Produit acheté en Septembre 2022...
aurélien	1.0	: 27 février 2023	2023-02-27	1.0	encore une commande partiellement annulée au d...	FR	encore une commande partiellement...
...

Analyse et exploration rapide du dataframe

Nous avons très peu de valeurs manquantes dans ce dataframe et peu de doublons en comparaison à la taille totale du fichier.

Nous allons procéder à leur suppression. Nous disposons au final pour ce dataset d'une bonne volumétrie et avec des informations complètes. ce qui est très utile et qualitatif pour la suite de notre analyse.

```
[3]: display(df.info())

display(df.isna().sum())

display(print('le nombre de doublons est:', df.duplicated().sum()))

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168897 entries, 0 to 168896
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   nom_client      168896 non-null  object
1   note_avis       168896 non-null  float64
2   date_achat      168896 non-null  object
3   date_avis       168896 non-null  object
4   nbr_avis        168895 non-null  float64
5   text_avis       168895 non-null  object
6   pays            168895 non-null  object
7   titre_avis      168893 non-null  object
dtypes: float64(2), object(6)
memory usage: 10.3+ MB

None

nom_client      1
note_avis       1
date_achat      1
date_avis       1
nbr_avis        2
text_avis       2
pays            2
titre_avis      4
dtype: int64

le nombre de doublons est: 337
```

Choix de l'intervalle de dates

Les avis récoltés sont postés depuis fin décembre 2014 jusqu'à février 2023 nous allons nous intéresser aux années civiles complètes dans le but de traiter des intervalles de temps similaires.. Donc, nous prendrons les dates du 01 janvier 2015 jusqu'au 31 décembre 2022. Ceci nous facilitera les comparaisons, l'étude des tendances et tenter de repérer des saisonnalités, si existantes, en fonction des années, des saisons, des mois.

Le nouveau dataset comporte alors **165967** entrées × 8 colonnes

Retraitement des commentaires

Avant de procéder à une étape de traitement des commentaires, nous avons créée des nouvelles variable à partir des commentaires, et des titres :

- 'longueur' qui mesure la longueur de chaque commentaire, en nombre de caractères.
- 'nb_mots' qui mesure le nombre de mots dans chaque commentaire.
- 'majuscule' qui mesure le nombre de caractères en majuscules pour chaque commentaire.
- 'ponct' qui mesure le nombre de points d'exclamation et d'interrogation dans chaque commentaire.

Des équivalents à ces 4 nouvelles variables ont été créées pour le titre des commentaires.

Ensuite, nous avons procédé à une étape de prétraitement des données (Commentaire et Titre) en utilisant la librairie NLTK (Natural Language Toolkit) pour nettoyer les données textuelles dans la variable 'Commentaire'. Nous avons appliqué les étapes suivantes : suppression des caractères spéciaux, mise en minuscules, suppression des stopwords (mots très courants qui ne portent pas de sens), et enfin, la lemmatisation (réduction des mots à leur forme de base). Cette étape est importante pour améliorer la qualité de nos données textuelles et faciliter l'extraction des informations importantes pour la prédiction.

Nous avons également réfléchi à introduire, avant la lemmatisation, une étape de correction orthographique. Notre test s'effectua avec la méthode spellchecker, mais le temps d'exécution était très très long (plusieurs semaines d'exécution estimées sur le jeu de données scrappées), ce malgré l'exécution du code sur plusieurs cœurs de notre processeur en simultané. Bien que pouvant être intéressant, voire important, l'intégration de spellchecker à notre code de nettoyage a été déclinée pour le moment, d'autres solutions sont à l'étude, notamment des tests avec d'autres méthodes, ou l'amélioration de notre puissance de calcul afin de parvenir à réaliser ces corrections dans un temps raisonnable.

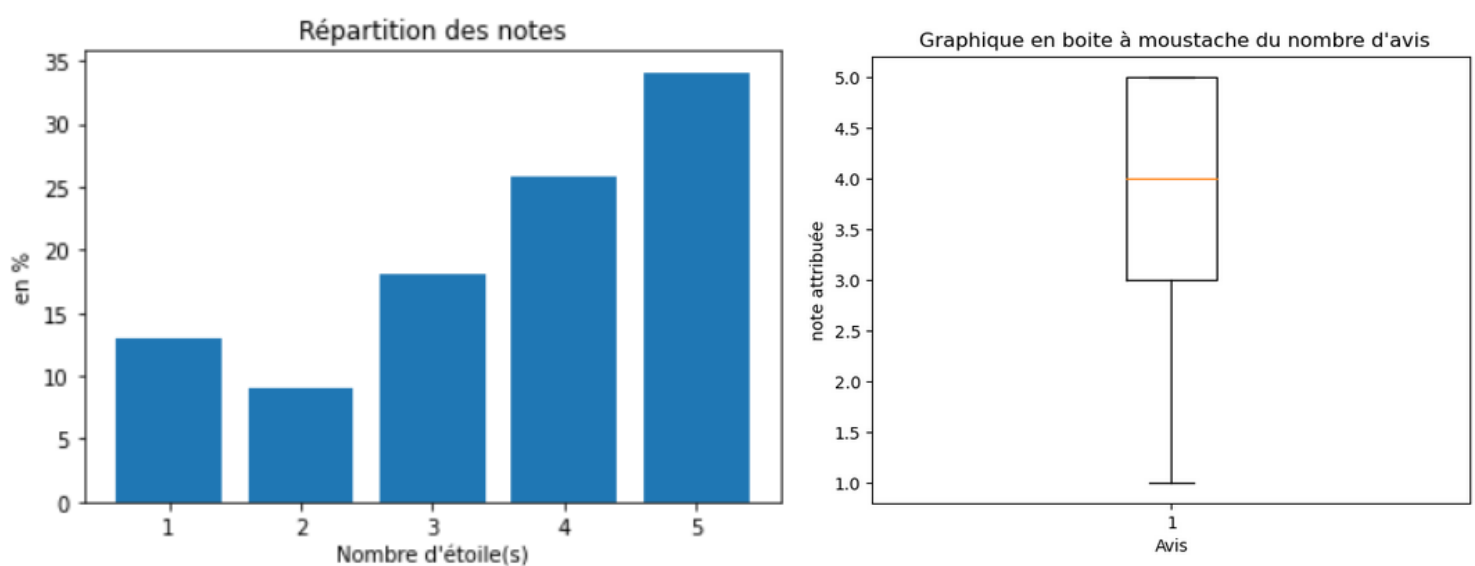
Concernant la normalisation ou la standardisation des données, nous n'avons pas jugé nécessaire de le faire pour la variable 'date', 'date' étant une variable temporelle. En fonction des performances des modèles à venir, nous ferons peut-être le choix de transformer la variable 'star' en variable binaire, avec un valeur pour les mauvaises notes, et une valeur pour les bonnes notes. Nous allons en revanche normaliser les variables 'ecart', 'longueur', 'longueur_titre' 'nb_mots',

'nb_mots_titre', 'majuscules', 'majuscules_titre', 'ponct', 'ponct_titre' et 'nb_avis' afin de réduire l'écart-type des variables.

Suite au nettoyage et à la visualisation des mots et ensemble de mots (ngrams) les plus fréquents parmi les commentaires et les titres, de nouvelles variables ont été créées, prenant les valeurs 0 ou 1 suivant la présence ou non des mots ou ensemble de mots sélectionnés, à l'intérieur des commentaires, ou des titres. Ces variables seront très utiles lors des premiers travaux de modélisation et de classification.

Visualisations et Statistiques

Observons les relations entre les différentes variables et la variable cible



Répartition de notre variable cible, le nombre d'étoiles

Selon ces deux graphiques, nous remarquons que les scores à 1* sont plus nombreux que les scores à 2*. ensuite de 2* à 5*, nous observons une certaine linéarité. Ceci est confirmé par le graphique en boîte à moustache et la répartition par quantile.

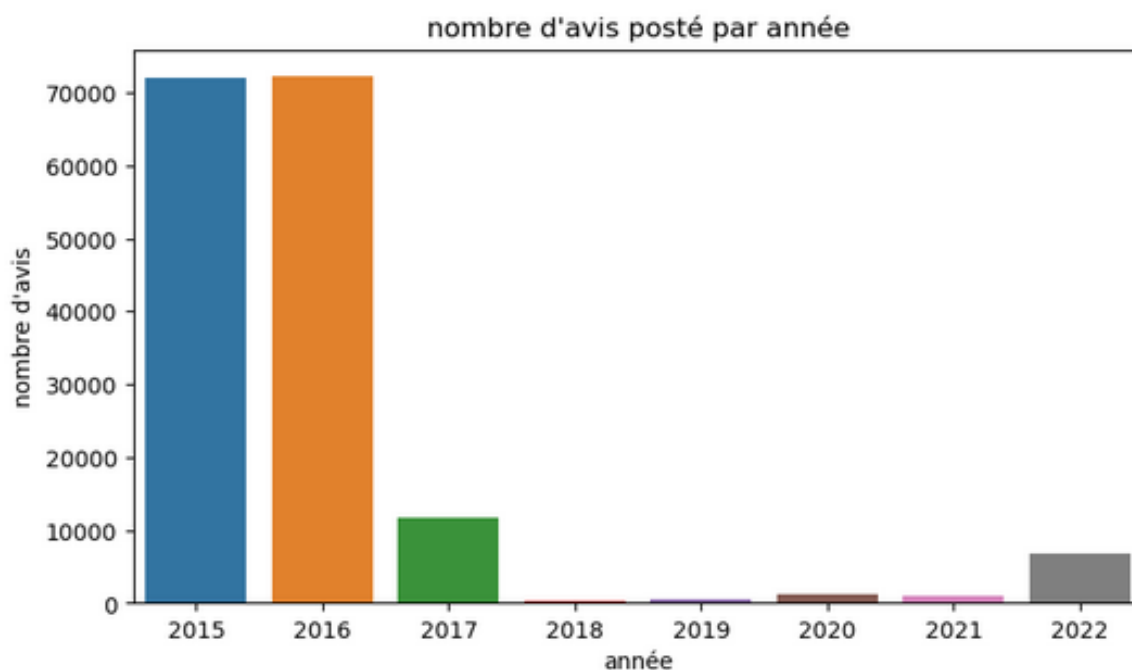
```
df['star'].value_counts(normalize=True)
```

5.0	0.341056
4.0	0.258348
3.0	0.180596
1.0	0.128997
2.0	0.091003

Répartition et distribution du nombre d'avis postés

Une première observation concerne le nombre d'avis publiés par an. Il est clairement visible que les années 2015 et 2016 ont connu de fortes publications de la part des clients. elles se distinguent des années suivantes.

pour les autres années, le nombre d'avis ne semble pas suivre une courbe ou une tendance, il est assez dispersé.

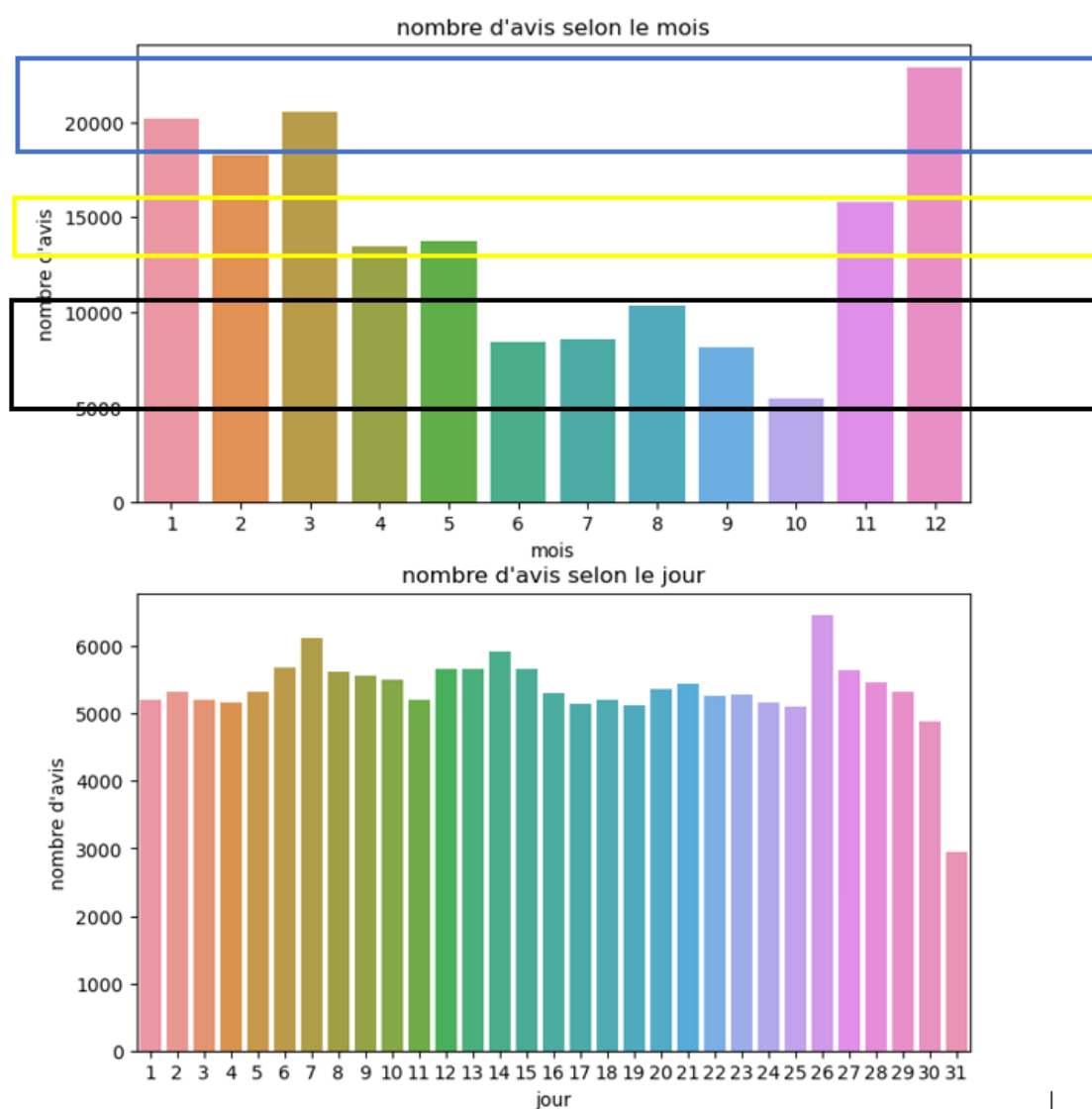


```
df['date_avis'].dt.year.value_counts()
```

```
2016    72261
2015    71925
2017    11727
2022     6843
2020     1212
2021     1089
2019       573
2018       337
```

```
Name: date_avis, dtype: int64
```

Graphiques des répartitions du nombre d'avis



La répartition des avis selon les mois montre globalement 3 périodes :

- Période de forte activité : On compte le plus de publications par mois et concerne les mois de janvier, février, mars et décembre

- Période de moyenne activité : cela concerne les mois de avril, mai et novembre
- Période de moindre activité : en juin, juillet, août, septembre et octobre

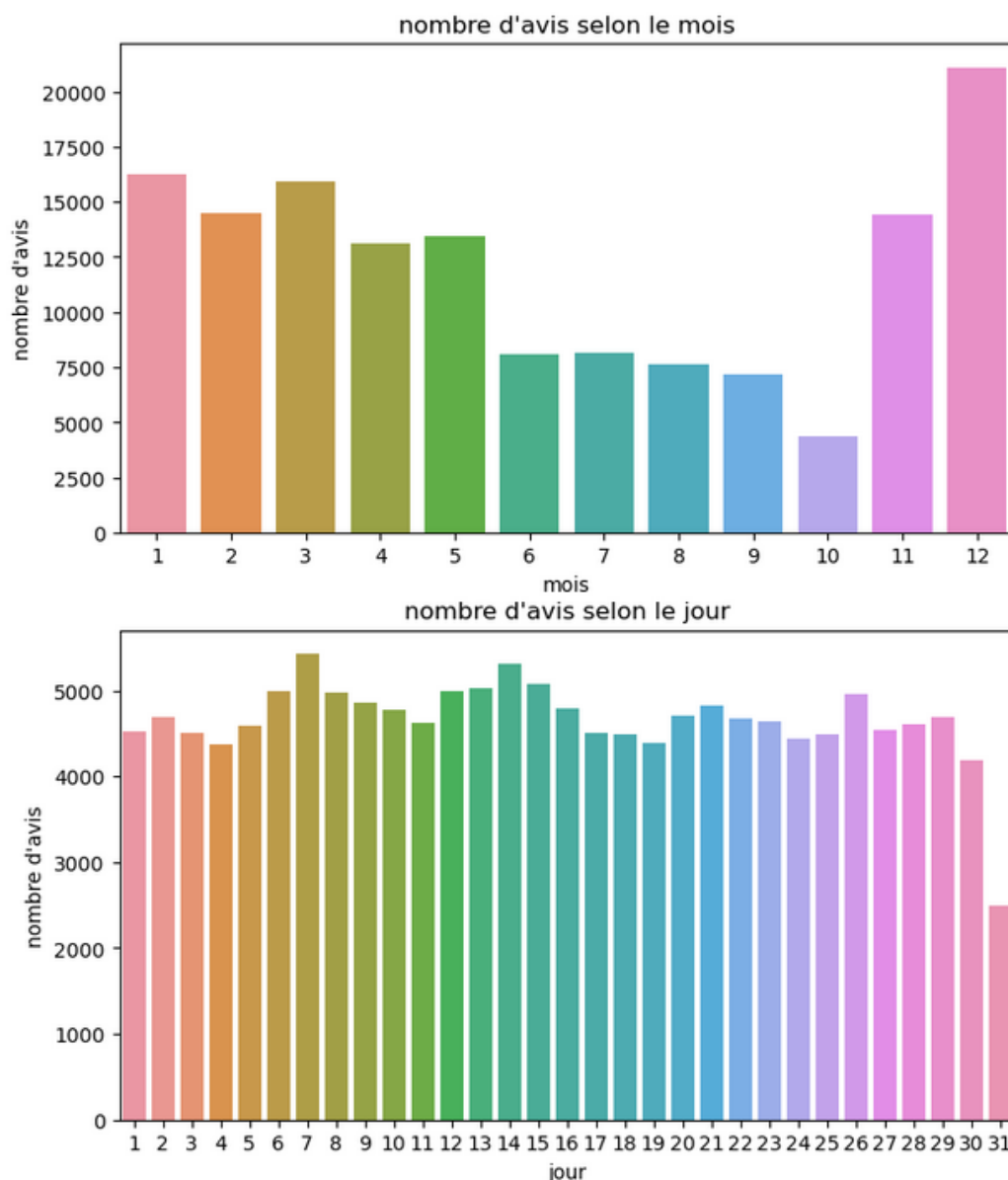
Au cours d'un mois, la distribution du nombre d'avis ne semble pas dégager une tendance particulière. Sur le graphique nous constatons que le jour numéro 31 est moins élevé que les autres, ceci est dû au fait que seuls 7 mois de l'année sont comptabilisés en jour 31.

Etude de la répartition pour l'année 2015 et 2016

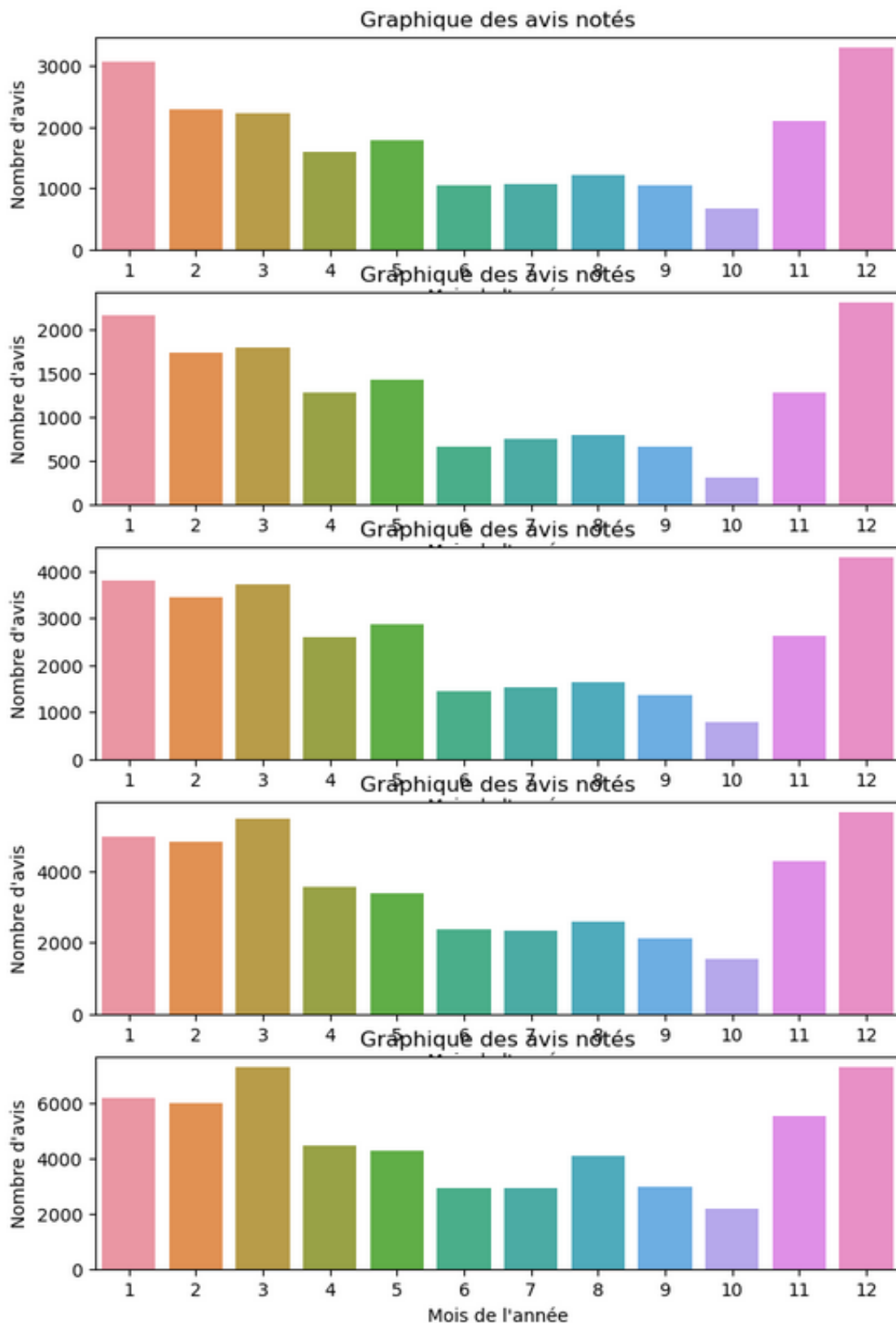
L'année 2015 et 2016 sont marquées par un nombre très élevé d'avis postés, 144 186 avis sur un total de **165967**, ce qui représente plus de 86% des entrées. Nous nous posons la question sur le poids de ces années dans notre analyse et future modélisation. Dans le graphique suivant, nous nous intéressons à la répartition des avis pour 2015, 2016.

Observation :

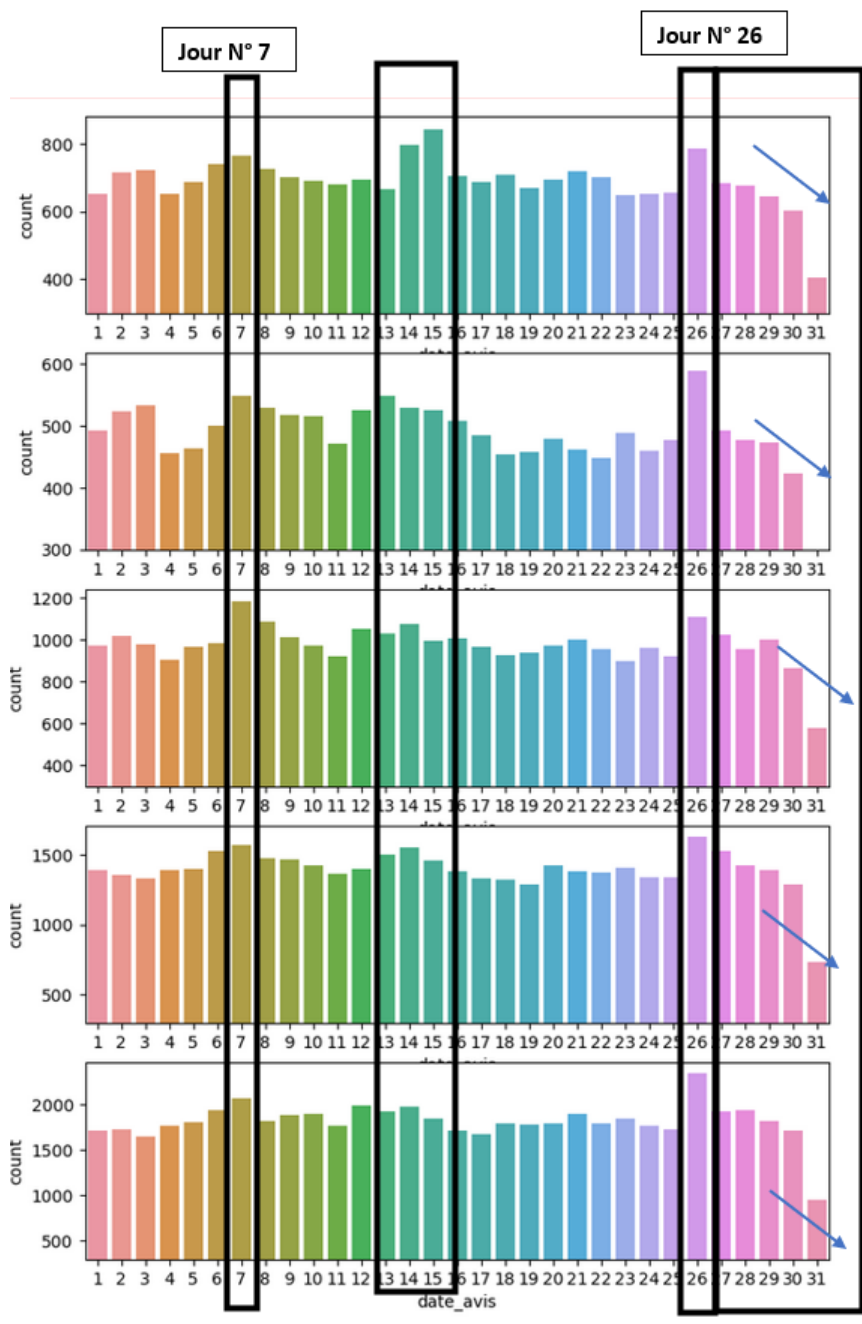
Nous remarquons à peu de chose près, que le nombre d'avis par mois et par jour suit le même graphique que précédemment.



Graphique de la répartition des avis de 1* à 5* selon les mois



Graphique de la répartition des avis de 1* à 5* selon les jours du mois



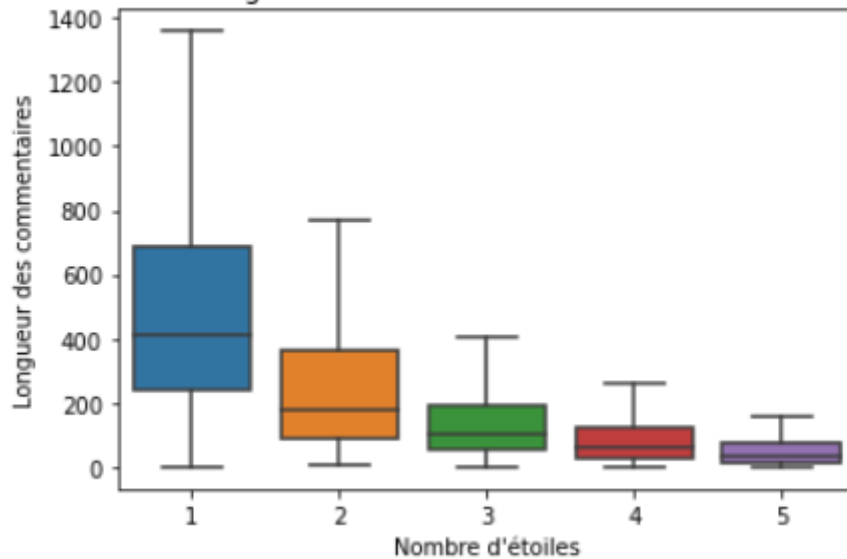
Visuellement, les avis allant de 1 à 5 suivent la même évolution tout au long des jours du mois. On observe :

- Une légère hausse des avis jusqu'au jour 7 qui détermine un pic (à explorer) suivi d'une légère baisse jusqu'à la mi de chaque mois.
- Une reprise des publication entre les 12-13-14-15 environ de chaque mois
- Un pic au jour 26 (à explorer) suivi d'une baisse du nombre d'avis

Remarque: Il est possible que ces pics apparaissent suite à des vagues de rappels de la part des entreprises et/ou de la plateforme à leurs clients de laisser des commentaires.

Relation entre “star” et les caractéristiques des commentaires

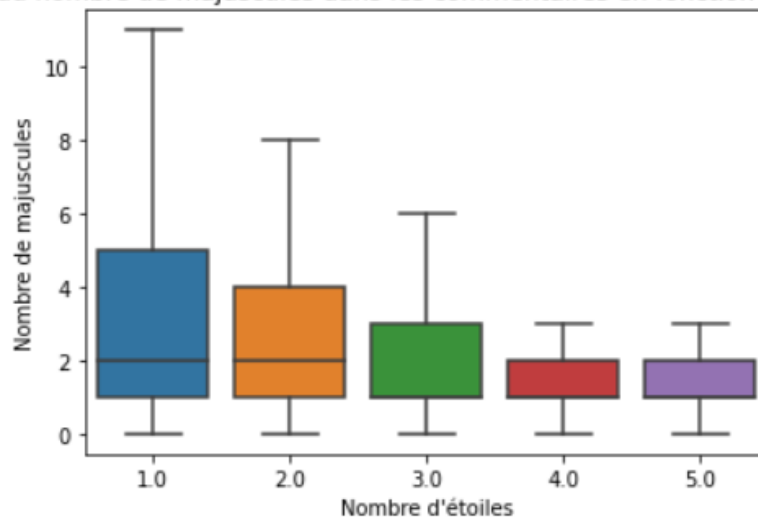
Distribution de la longueur des commentaires en fonction du nombre d'étoiles



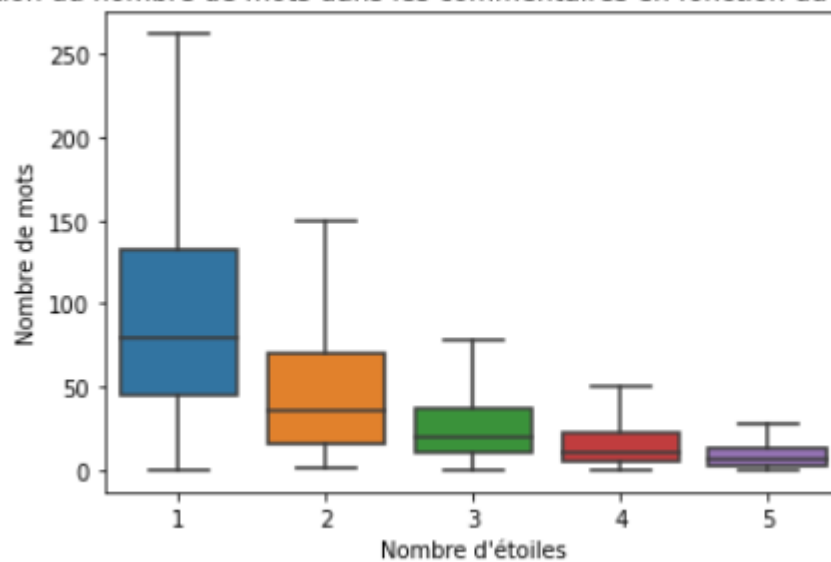
A l'analyse visuelle du dataset, nous avons identifié que plus le commentaire est long, plus il y a de chances que la note associée soit négative. Ce qui se vérifie par le coefficient de corrélation négatif plus bas.

Nous remarquons les mêmes phénomènes concernant le nombre de mots (variables très liées à la longueur des commentaires), les nombres de majuscules et les certains caractères de ponctuation.

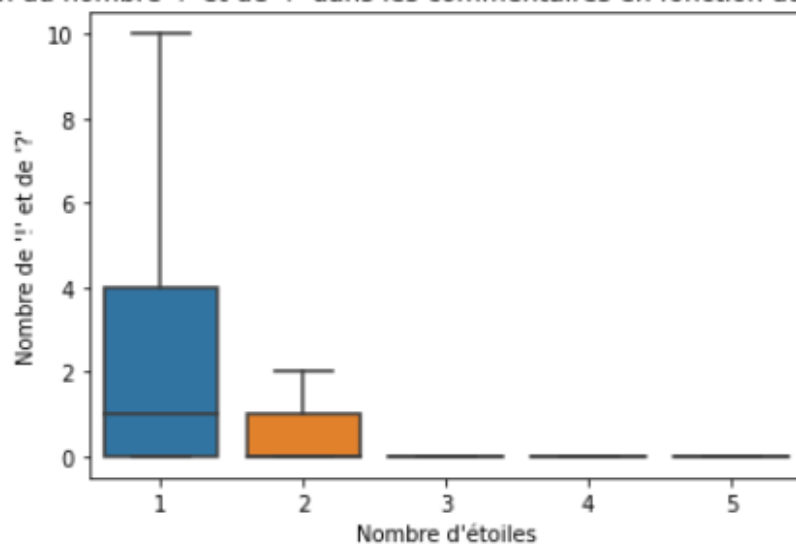
Distribution du nombre de majuscules dans les commentaires en fonction du nombre d'étoiles



Distribution du nombre de mots dans les commentaires en fonction du nombre d'étoiles



Distribution du nombre '!' et de '?' dans les commentaires en fonction du nombre d'étoiles

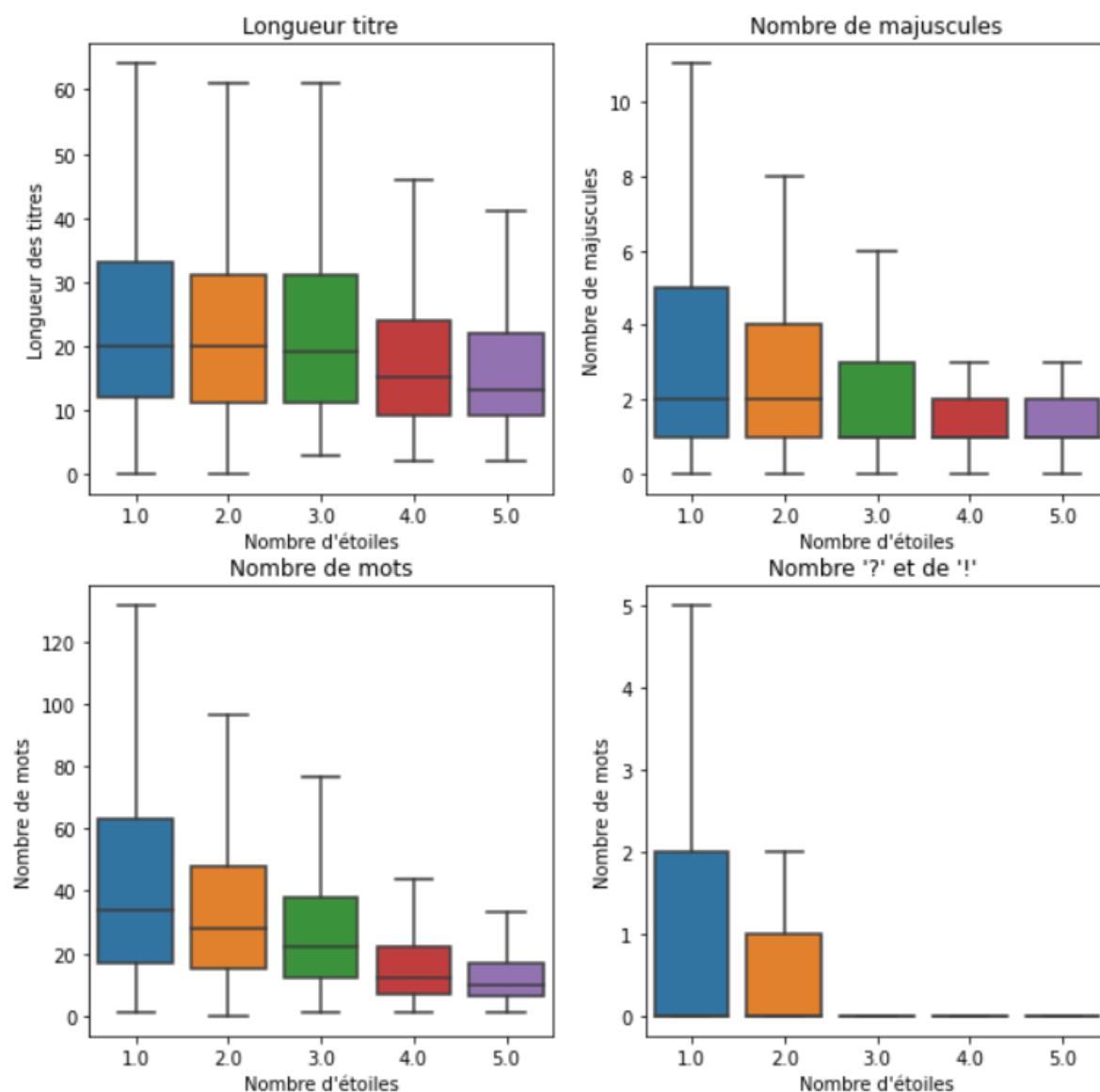


Matrice de corrélation :

```
1 df[["star", "longueur", "majuscule", "ponct", "nb_mots"]].corr()
```

	star	longueur	majuscule	ponct	nb_mots
star	1.000000	-0.405823	-0.103718	-0.208811	-0.415711

Des observations similaires peuvent être faites concernant les titres des commentaires, mais de manière moins marquée, sans doute en raison de la limitation du nombre de caractères dans les titres.



Matrice de corrélation :



```
1 df[["star", "longueur_titre", "majuscule_titre", "ponct_titre", "nb_mots_titre"]].corr()
```

	star	longueur_titre	majuscule_titre	ponct_titre	nb_mots_titre
star	1.000000	-0.182322	-0.029035	-0.048556	-0.174929

Analyse des commentaires



A l'issue du nettoyage et de la lemmatisation des commentaires, des wordclouds ont été réalisés afin de représenter la fréquence des mots et des ensembles de mots (ngrams) les plus fréquents. Le plus intéressant étant de cibler les mots et ensemble de mots les plus fréquents pour les commentaires/titres négatifs (1, 2 et 3 étoiles), en parallèle avec les mots les plus fréquents des commentaires/titres positifs (4 et 5 étoiles).

WordCloud des commentaires, les mots seuls :

Mauvaises notes		Mot/litération commande 29600 livraison 21155 colis 18265 plus 17652 très 15066 article 11347 reçu 11055 produit 9801 site 9272 produits 9270 bien 8537 trop 8455 fois 8229 commandé 7920 tout 6953 service 6801 qualité 6520 déçue 6383 fait 5935 client 5825
Bonnes notes		Mot/litération livraison 27404 très 27332 commande 26621 bien 17122 site 14954 produits 12376 produit 11536 bon 11171 qualité 10236 tout 9521 délais 9500 prix 8931 satisfaite 8875 peu 7851 conforme 7752 temp 7417 long 7383 colis 7354 article 7238 délai 7108

Les mots seuls donnent une première information, mais on retrouve aussi des mots communs aux bonnes et mauvaises notes. Il est nécessaire d'observer les n-grams pour avoir plus d'informations sur le sens des mots.

WordCloud des commentaires, 2-grams :

Mauvaises notes		<p>Mots/litération</p> <p>('service', 'client') 3737 ('point', 'relais') 3218 ('trop', 'long') 2872 ('délai', 'livraison') 2193 ('très', 'déçue') 2189 ('frais', 'port') 2131 ('délais', 'livraison') 2078 ('livraison', 'trop') 1766 ('cette', 'commande') 1516 ('très', 'déçu') 1384 ('geste', 'commercial') 1344 ('première', 'fois') 1312 ('beaucoup', 'trop') 1237 ('nai', 'reçu') 1131 ('mondial', 'relay') 1113 ('j'avais', 'commandé') 1055 ('mauvaise', 'qualité') 1018 ('plus', 'tard') 951 ('retard', 'livraison') 950 ('reçu', 'colis') 944</p>
Bonnes notes		<p>Mots/litération</p> <p>('très', 'bien') 4793 ('très', 'bon') 3752 ('délai', 'livraison') 3706 ('délais', 'livraison') 3419 ('très', 'satisfaite') 3226 ('bonne', 'qualité') 3146 ('peu', 'long') 3039 ('rien', 'dire') 2776 ('livraison', 'peu') 2472 ('bien', 'passé') 2412 ('date', 'lexpérience') 2204 ('bon', 'site') 2167 ('produit', 'conforme') 2164 ('trop', 'long') 1894 ('bon', 'produit') 1872 ('rien', 'redire') 1858 ('livraison', 'rapide') 1738 ('très', 'contente') 1712 ('tout', 'bien') 1706 ('qualité', 'prix') 1705</p>

Avec les 2-grams, nous pouvons mieux identifier si le commentaire est positif ou non. Par exemple, les mots associés au mot “très” permettent de bien catégoriser les avis. On oppose bien “très satisfaite” à “très déçue” par exemple. Par contre, certains duos restent flous, comme “délai livraison”.

WordCloud des commentaires, 3-grams :



Mauvaises notes		<p>Mots/Itération</p> <p>('livraison', 'trop', 'long') 951 ('beaucoup', 'trop', 'long') 619 ('plus', 'dun', 'mois') 523 ('livraison', 'beaucoup', 'trop') 505 ('délai', 'livraison', 'trop') 442 ('livraison', 'trop', 'longue') 419 ('délais', 'livraison', 'trop') 410 ('nest', 'première', 'fois') 359 ('très', 'mauvaise', 'qualité') 350 ('aucun', 'geste', 'commercial') 335 ('service', 'après', 'vente') 311 ('autre', 'point', 'relais') 274 ('entre', 'commande', 'livraison') 243 ('cette', 'fois', 'ci') 220 ('déçue', 'cette', 'commande') 217 ('colis', 'point', 'relais') 217 ('livraison', 'point', 'relais') 217 ('nai', 'toujours', 'reçu') 216 ('livraison', 'très', 'long') 212 ('jours', 'plus', 'tard') 210</p>
Bonnes notes		<p>Mots/Itération</p> <p>('très', 'bon', 'site') 1246 ('livraison', 'peu', 'long') 1200 ('tout', 'bien', 'passé') 1145 ('rapport', 'qualité', 'prix') 1009 ('très', 'bonne', 'qualité') 988 ('tout', 'très', 'bien') 898 ('très', 'bien', 'passé') 858 ('délai', 'livraison', 'peu') 695 ('délai', 'livraison', 'respecté') 691 ('très', 'bon', 'produit') 686 ('livraison', 'peu', 'longue') 672 ('bon', 'rapport', 'qualité') 636 ('livraison', 'trop', 'long') 636 ('très', 'satisfaite', 'commande') 494 ('délais', 'livraison', 'peu') 493 ('entre', 'commande', 'livraison') 447 ('délais', 'livraison', 'respectés') 413 ('produit', 'conforme', 'description') 390 ('date', 'l'expérience', 'août') 389 ('très', 'bon', 'état') 340</p>

Avec les 3grams, on obtient les détails manquants sur certains termes, notamment “délai livraison”, qui prend la forme “délai livraison respecté” ou “délai livraison trop”...longue.

Analyse des titres


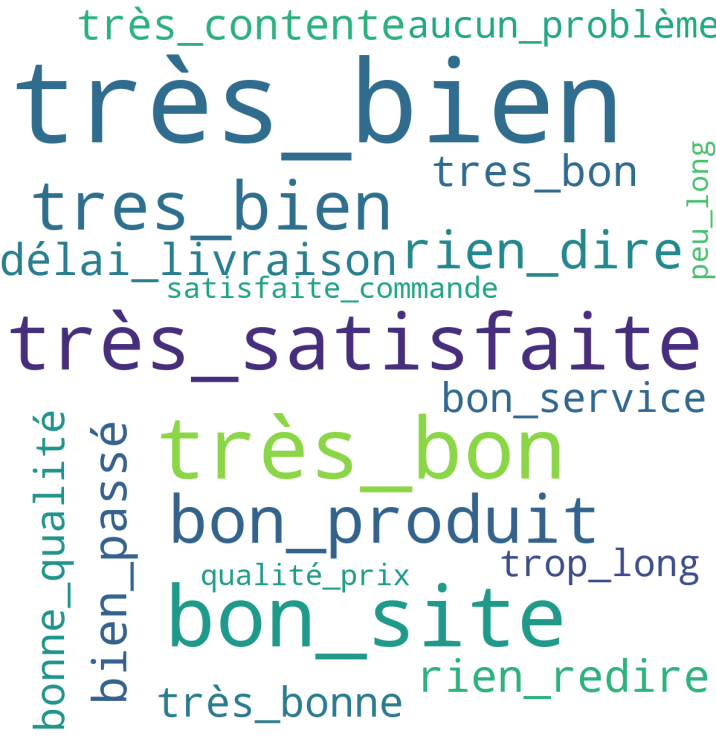
La même analyse peut être réalisée sur les titres des commentaires

WordCloud des titres, les mots seuls :

Mauvaises notes		Mot/Itération livraison 10078 commande 7297 très 3669 trop 3659 produit 3434 déçue 3326 colis 3244 non 3058 long 2458 déçu 2376 article 2324 qualité 2178 délai 1883 produits 1848 retard 1753 reçu 1747 bien 1645 mauvaise 1469 erreur 1357 délais 1350
Bonnes notes		Mot/Itération très 14117 bien 12814 commande 9688 satisfaite 7806 bon 7643 parfait 7426 site 5948 livraison 5847 super 4713 excellent 3836 tres 3578 produit 3529 rapide 2749 bonne 2697 conforme 2627 qualité 2240 produits 2204 rien 2184 tout 2183 satisfait 1762



Le titre doit être clair et concis, ce qui facilite le repérage de mots récurrents. Ainsi, à l'inverse des commentaires, on retrouve rapidement des mots positifs et négatifs, compréhensifs pour la plupart, tout seul.

WordCloud des titres, 2-grams :

Mauvaises notes		<p>Mots/litération</p> <ul style="list-style-type: none"> ('trop', 'long') 1670 ('livraison', 'trop') 1344 ('délai', 'livraison') 1132 ('délais', 'livraison') 748 ('non', 'conforme') 747 ('très', 'déçue') 713 ('trop', 'longue') 674 ('commande', 'incomplète') 590 ('mauvaise', 'qualité') 578 ('retard', 'livraison') 567 ('commande', 'non') 517 ('livraison', 'non') 469 ('très', 'déçu') 456 ('service', 'client') 453 ('mieux', 'faire') 388 ('frais', 'port') 346 ('point', 'relais') 334 ('problème', 'livraison') 333 ('non', 'respecté') 315 ('bon', 'produit') 299
Bonnes notes		<p>Mots/litération</p> <ul style="list-style-type: none"> ('très', 'bien') 5328 ('bon', 'site') 2543 ('très', 'bon') 2514 ('très', 'satisfaite') 2138 ('tres', 'bien') 1616 ('bon', 'produit') 1393 ('rien', 'dire') 919 ('bien', 'passé') 750 ('rien', 'redire') 683 ('délai', 'livraison') 658 ('très', 'contente') 620 ('tres', 'bon') 605 ('bonne', 'qualité') 582 ('bon', 'service') 573 ('aucun', 'problème') 560 ('qualité', 'prix') 552 ('très', 'bonne') 548 ('tout', 'bien') 507 ('peu', 'long') 504 ('trop', 'long') 499

Comme pour les commentaires, c'est principalement sur les termes "délais livraisons" qu'il est nécessaire d'avoir une analyse de niveau 3

WordCloud des commentaires, 3 grams :

Mauvaises notes		Mots/Itération ('livraison', 'trop', 'long') 634 ('livraison', 'trop', 'longue') 458 ('délai', 'livraison', 'trop') 291 ('peut', 'mieux', 'faire') 220 ('délais', 'livraison', 'trop') 207 ('livraison', 'non', 'respecté') 189 ('produit', 'non', 'conforme') 174 ('délai', 'trop', 'long') 146 ('commande', 'non', 'conforme') 138 ('livraison', 'beaucoup', 'trop') 137 ('délai', 'livraison', 'non') 131 ('beaucoup', 'trop', 'long') 128 ('très', 'mauvaise', 'qualité') 125 ('non', 'conforme', 'commande') 109 ('peu', 'mieux', 'faire') 109 ('commande', 'non', 'reçue') 85 ('colis', 'non', 'reçu') 81 ('délais', 'livraison', 'non') 81 ('délai', 'livraison', 'trop') 78 ('livraison', 'non', 'conforme') 73
Bonnes notes		Mots/Itération ('très', 'bon', 'site') 1206 ('rapport', 'qualité', 'prix') 417 ('très', 'bon', 'produit') 401 ('tout', 'bien', 'passé') 367 ('bon', 'rapport', 'qualité') 322 ('très', 'bien', 'passé') 268 ('très', 'bon', 'site') 262 ('bon', 'traitement', 'commande') 256 ('tout', 'très', 'bien') 254 ('livraison', 'trop', 'long') 230 ('très', 'bon', 'service') 215 ('livraison', 'peu', 'long') 162 ('très', 'bonne', 'qualité') 158 ('très', 'satisfaite', 'commande') 156 ('commande', 'sans', 'problème') 156 ('livraison', 'trop', 'longue') 144 ('livraison', 'peu', 'longue') 137 ('très', 'bon', 'produit') 116 ('délai', 'livraison', 'trop') 116 ('très', 'bon', 'rapport') 107

Les mots et ensemble de mots les plus présents, marquants, seront retenus pour créer des nouvelles variables qui prendront 0 ou 1 en fonction de leur présence ou non dans les titres et/ou commentaires.

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

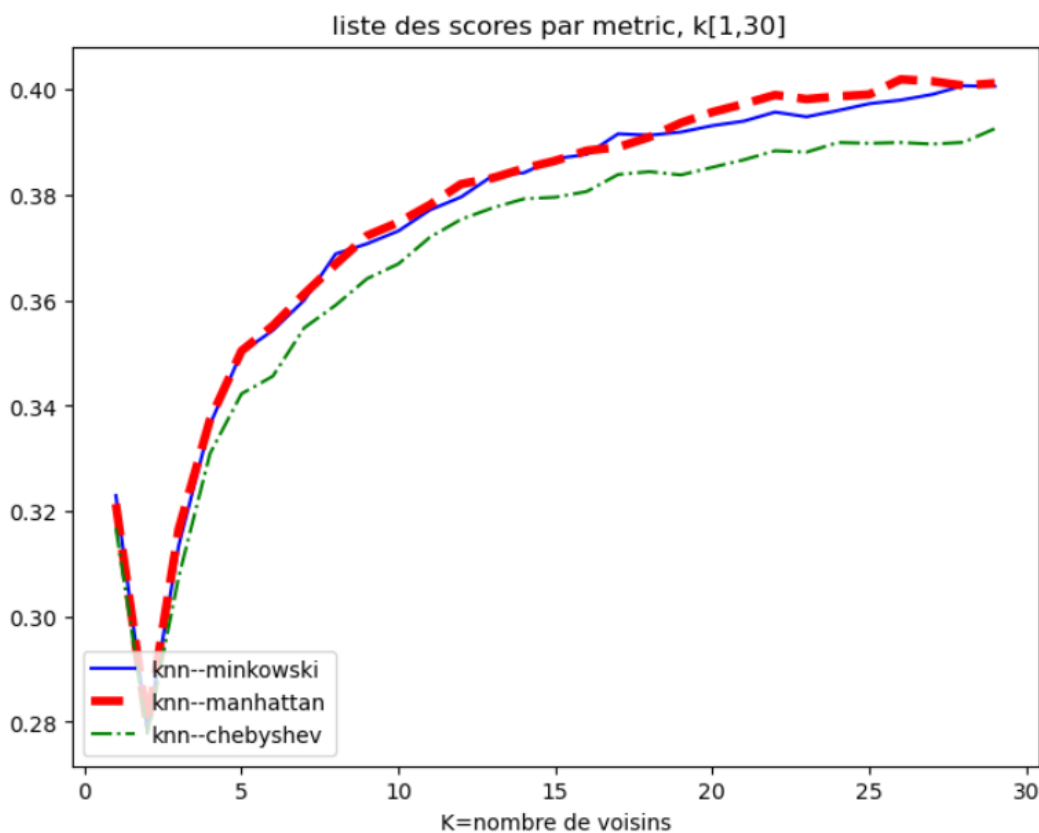
Nous nous sommes dans un premier temps intéressés au modèle Knn des plus proches voisins. Nous avons étudié, en premier lieu, les métadonnées de base de "data_preparee.csv" qui comportent, les longueurs du texte, nombre de signes de ponctuation, majuscules,

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.

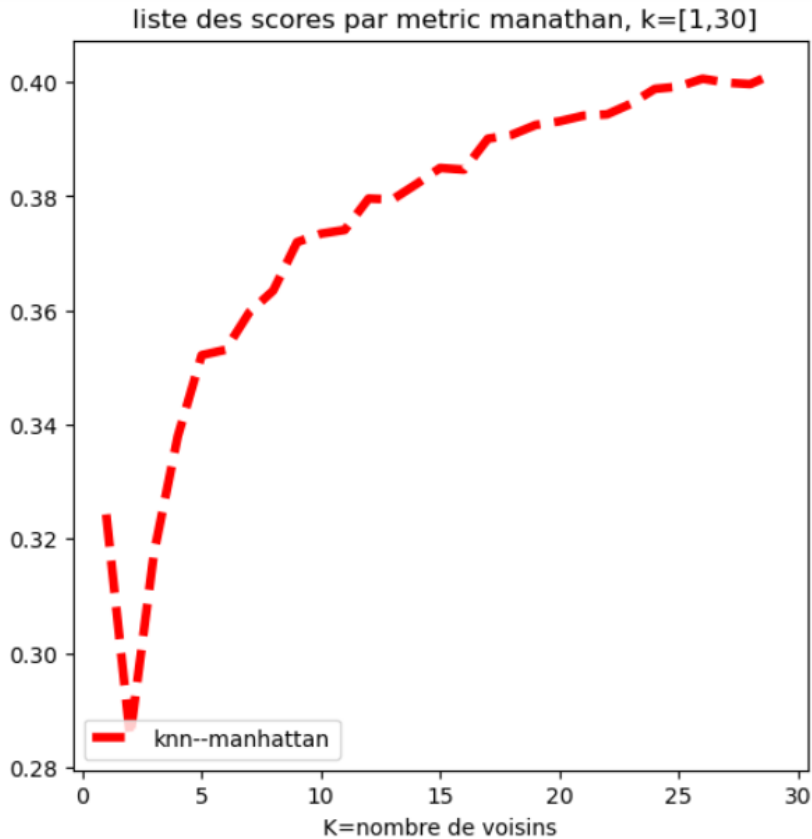
Réduction de dimensions :

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'information. Nous nous sommes intéressés au 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
cas 2	Négatif	Négatif	Neutre	Positif	Positif
cas 3	Négatif	Négatif	Négatif	Positif	Positif
cas 4	Négatif	Négatif	Négatif	Négatif	Positif
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

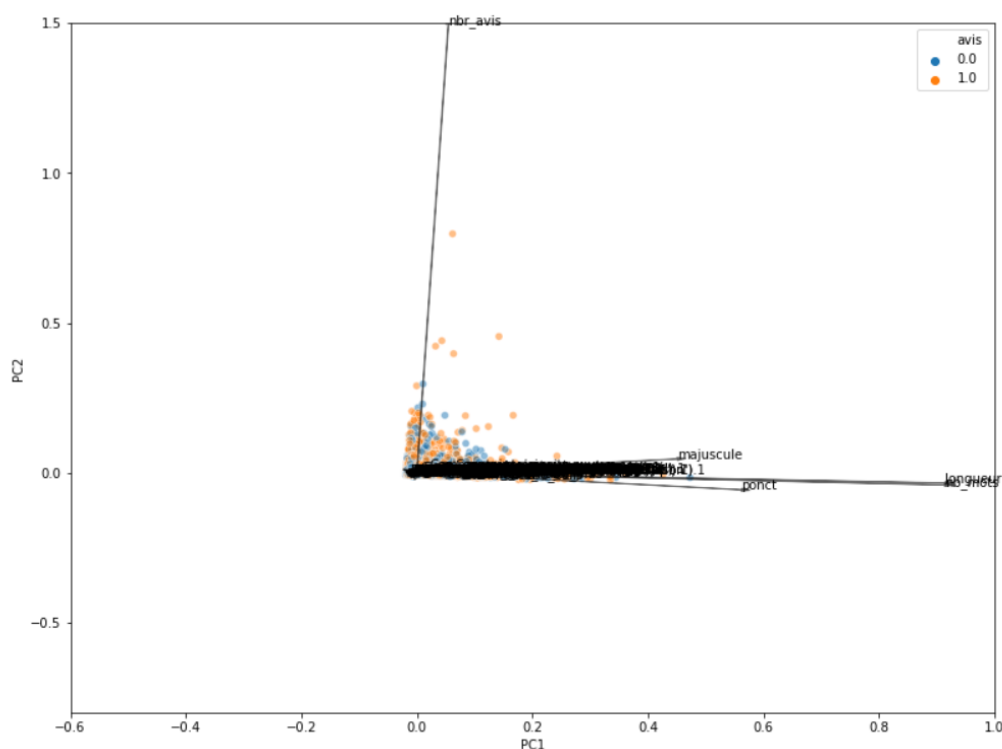
Les modèles effectués avec un Bag of Word

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 xgbooost 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

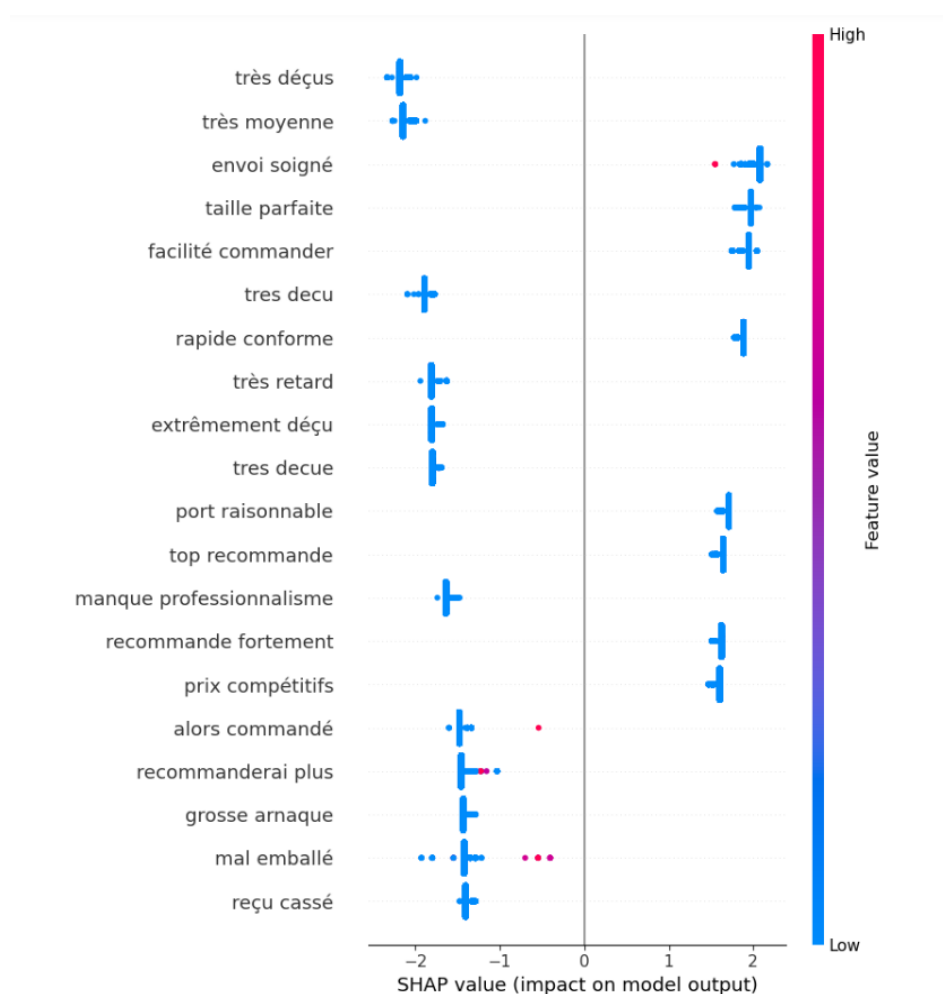
Le traitement par TF-IDF est ici assez peu concluant, ou en tous les cas n'apporte pas de précision supplémentaire.

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 supérieure 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 second 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

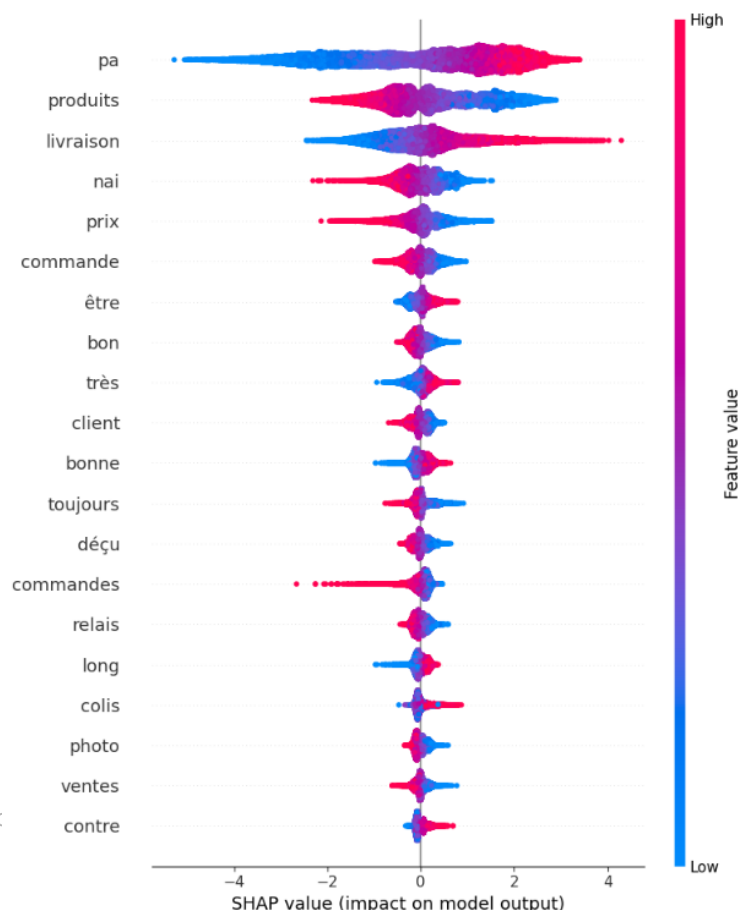
Après plusieurs ajustements, un dernier test a été réalisé avec un extrême gradient boosting, donc les hyper-paramètres ont été ajustés avec un Grid Search. Plusieurs tests ont été réalisés autour de réglages du learning-rate [1, 0.5, 0.3, 0.25, 0.225, 0.2, 0.175, 0.15, 0.1, 0.05], de la profondeur maximale [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

Difficultés rencontrées lors du projet

Le projet a eu comme intérêt de nous faire travailler un aspect assez peu étudié lors de la formation : l'analyse de sentiment, et les prémices de la construction de modèles de compréhension du langage.

Ce projet avait comme spécificité l'élaboration du dataset à partir du scrapping d'un site internet. Cette première étape nous a un peu ralenti.

En effet, le projet a nécessité l'étude anticipée de plusieurs modules de formation, notamment ceux sur le web scraping (BeautifulSoup et Selenium) et le module sur le Text Mining. Certaines pistes de construction de modèles sont arrivées un peu en décalage par rapport au calendrier initial d'avancement du projet.

L'accès et le temps d'acquisition des connaissances liés au scraping a ralenti le véritable démarrage du projet, le scrapping étant une étape préalable au nettoyage, l'analyse, et la préparation du jeu de données en vue de réaliser des modélisations.

Il est nécessaire de signaler que nous avons également été ralenti par rapport au planning initial en raison de la non-participation au projet de deux camarades de la promotion. Cette absence s'est traduite par une charge de travail répartie sur 2 personnes au lieu de 4, et la réduction de l'émulsion intellectuelle autour du projet.

Le spectre du projet a été un peu réduit en conséquence, limitant le modèle à de l'analyse de sentiment, là où il aurait pu être poussé vers de l'identification et le paramétrage de réponses automatiques à des commentaires pointants vers des problèmes spécifiques identifiés.

Il est à noter que nous avons également rencontré des limitations techniques, en termes de mémoire vive et de puissance de calcul. Après avoir passé un peu de temps à essayer de paramétrer un environnement python avec accélération matériel par carte graphique, nous nous sommes finalement tournés vers les services de Google via Colab en version Pro, afin d'accéder à des sessions avec plus de RAM et de meilleurs processeurs graphiques. Nous avons pu, grâce à cela, aller plus loin qu'initialement dans le tuning de nos modèles.

Bilan

Nous avons abouti à un modèle d'analyse de sentiments, permettant de classer si un commentaire laissé est positif ou négatif. Nous nous sommes éloignés de l'objectif de classification des commentaires sur une échelle de 1 à 5, les modèles testés donnant de mauvais résultats sur les classes intermédiaires. Dépasser les contraintes techniques que nous avons rencontré aurait permis des modèles sur davantage de commentaires, de constituer des modèles initiales à davantage de dimensions, ce qui aurait permis, peut-être d'améliorer les prédictions, et de réaliser des classifications sur une échelle non-binaire.

Ce type de projet peut avoir une utilité pour, à la fois, détecter dans un volume important de données, les éléments d'insatisfaction de clientèles, soigner les réponses apportées aux commentaires, et tenter de réduire les éléments d'insatisfaction. C'est le type de modèle qui sert également en première instance aux ChatBot. Pour avoir une réelle utilité métier, le modèle nécessiterait d'être davantage entraîné.

Le modèle actuel obtient un score de 87% de bonnes prédictions sur l'ensemble de test, avec une légère capacité à mieux détecter les commentaires positifs :

	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

Références

B. Gupta, M. Negi, K. Vishwakarma, G. Rawat and P. Badhani, "Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python", International Journal of Computer Applications (0975 – 8887) Volume 165 – No.9, May 2017

S. A. A. Hridoy, M. T. Ekram, M. S. Islam, F. Ahmed and R. M. Rahman, "Localized twitter opinion mining using sentiment analysis", Springer Open Journal (1186, 40165-015-0016-4), 2015

M.D. Devika, C. Sunitha and A. Ganesh, "Sentiment analysis: a comparative study on different approaches", Elsevier (87, 44–49), 2016

Pierre Megret, "Gensim Word2Vec Tutorial", 2018,
<https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial>