Supply Chain - Satisfaction des clients

Rapport 1 : rapport d'exploration, de data visualisation et de pre-processing des données

Leonard Heyerdahl
Alexandre PRZYBYLSKI
Alexis Garatti
Huazhen Hou

Objectif principal du projet :

Extraire des informations ainsi que les sentiments contenus dans des commentaires.

Objectifs spécifiques :

- ***Prédire la satisfaction d'un client** : problème de régression (prédire le nombre d'étoiles). Entraînement supervisé possible.*
- ***Identifier les points clés des avis** : localisation, nom d'entreprise...*
- ***Extraire les propos du commentaire et trouver les mots importants** (problème de livraison, article défectueux...) : approche non supervisé avec CamemBert.*
- ***Trouver une réponse rapide adaptée pour répondre au commentaire** (par exemple sur les reviews Google).*

Collecte des données

Sur TrustPilot France, nous avons choisi la catégorie Banques pour le volume estimé des données et pour une distribution des modalités (étoiles) relativement plus équilibrée que dans d'autres catégories. En effet lors d'un premier test sur la catégorie immobilier nous nous sommes aperçus qu'il y avait une prédominance des 4 et 5 étoiles.

TrustPilot ne met pas à disposition d'API, nous avons donc collecté les données par webscrapping. La stratégie de scraping pour Trustpilot a impliqué un processus en deux étapes pour collecter de manière exhaustive les données des entreprises et les avis dans les catégories sélectionnées.

La première phase, **le Scrap 1**, s'est concentrée sur l'extraction d'informations clés des entreprises à partir de la page de listes des banques. Les données ciblées pour le scraping comprenaient le nom des entreprises, la moyenne des étoiles attribuées par les utilisateurs, le nombre total d'avis et le lien direct vers la page de l'entreprise qui contient tous les avis. L'aboutissement de cette phase a été le stockage des données extraites dans un fichier CSV nommé "liste_entreprises". Le **Scrap 2** a utilisé la liste compilée des entreprises pour se pencher sur les pages individuelles de chaque entreprise. Ici, le scraping était plus détaillé, ciblant les avis des utilisateurs et leurs détails. Le scraper a été conçu pour recueillir le nom d'utilisateur, leur nombre d'avis, la localisation, la date de publication de l'avis, le titre de l'avis, le contenu de l'avis, et la date de l'expérience évaluée. Chaque information a été méticuleusement extraite pour former un ensemble de données détaillé des avis correspondant aux entreprises listées dans la phase de scraping précédente.

Les pages d'avis contenaient chacun 20 avis et nous avons adapté le scrapper pour qu'il détermine le nombre de pages au total et qu'il change la clé de page dans l'url d'extraction sur chaque itération du scrapper.
De manière préventive un délais de deux secondes a été appliqué entre chaque page pour éviter de spammer Trustpilot et que le scrapper soit bloqué par le site.
Nous avons sauvegardé un csv par entreprise, puis nous avons compilé les différents CSV en une base de données unique.

Résultat du scrapping :
Au total, nous avons recueilli 170751 avis sur les banques, en Français.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170751 entries, 0 to 170750
Data columns (total 11 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Unnamed: 0.1     170751 non-null  int64
 1   Unnamed: 0       170751 non-null  object
 2   user             170157 non-null  object
 3   etoiles          170163 non-null  object
 4   n_avis           169488 non-null  float64
 5   localisation     169487 non-null  object
 6   date_avis        169488 non-null  object
 7   titre_avis       169488 non-null  object
 8   text_avis        169488 non-null  object
 9   date_experience  168813 non-null  object
 10  page             168813 non-null  object
dtypes: float64(1), int64(1), object(9)
memory usage: 14.3+ MB
```

Nettoyage des données

Nous avons commencé par drop.na avant de drop les deux colonnes Unnamed (celles-ci contenaient uniquement le numéro de l'avis).

Afin de facilier la lecture du df, nous avons renommé la colonne page en Société (comme le scrapping a été fait à partir des pages des entreprises, la colonne avait ce nom de base).

Après quelques représentation des différentes variables, nous avons décidé de drop la colonne localisation. Etant donné que nous étions sur TrustPilot FR, nous avions plus de 99% d'avis en France (cf graph dans la partie Analyse Exploratoire).

Pour facilier les différentes représentations graphiques, nous avons ensuite attaqué la conversion des types de nos variables :

- Les étoiles d'object en int64 (de 1 à 5) avec pd.to_numeric

- Titre_avis et text_avis d'object en str
- La transformation de Date_experience et date_avis en datetime nous a demandé plus de code. Il y avait une partie str à enlever ainsi que la conversion des mois de français à anglais afin d'utiliser la commande to_datetime. Nous avons donc supprimé la partie str inutile avant de créer une fonction utilisant un dictionnaire pour transformer les mois français en mois anglais

Nous terminons ainsi avec le df_clean suivant :

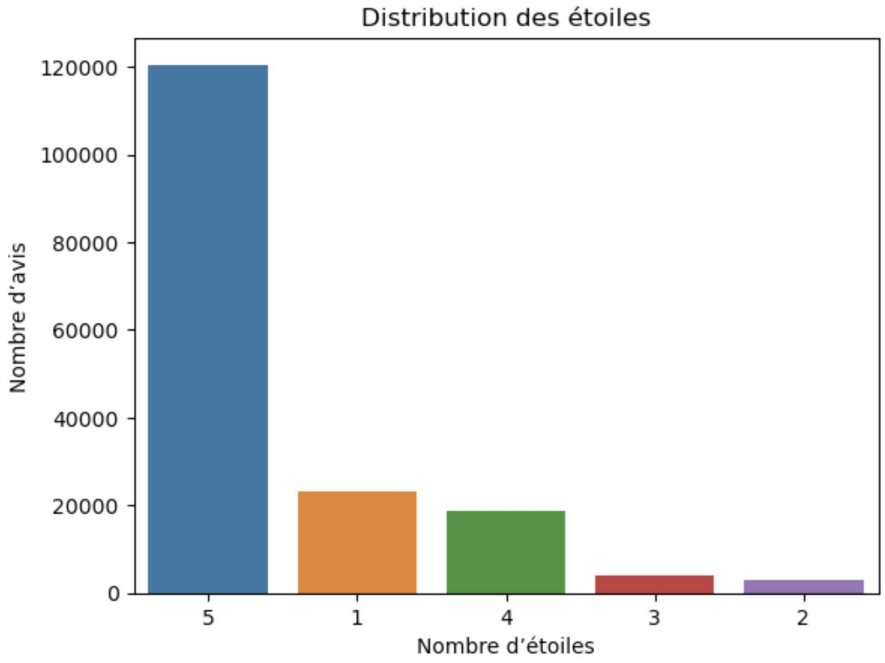
```
<class 'pandas.core.frame.DataFrame'>
Index: 168806 entries, 0 to 170750
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   user             168806 non-null  object
 1   etoiles          168806 non-null  int64
 2   n_avis           168806 non-null  float64
 3   date_avis        168806 non-null  datetime64[ns, UTC]
 4   titre_avis       168806 non-null  object
 5   text_avis        168806 non-null  object
 6   date_experience  168806 non-null  datetime64[ns]
 7   Société          168806 non-null  object
dtypes: datetime64[ns, UTC](1), datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 11.6+ MB
```

Analyse exploratoire des données

Après ce cleaning, nous avons commencé à analyser nos différentes données :

Les 5 étoiles sont prédominantes dans le jeu de données en globalité :

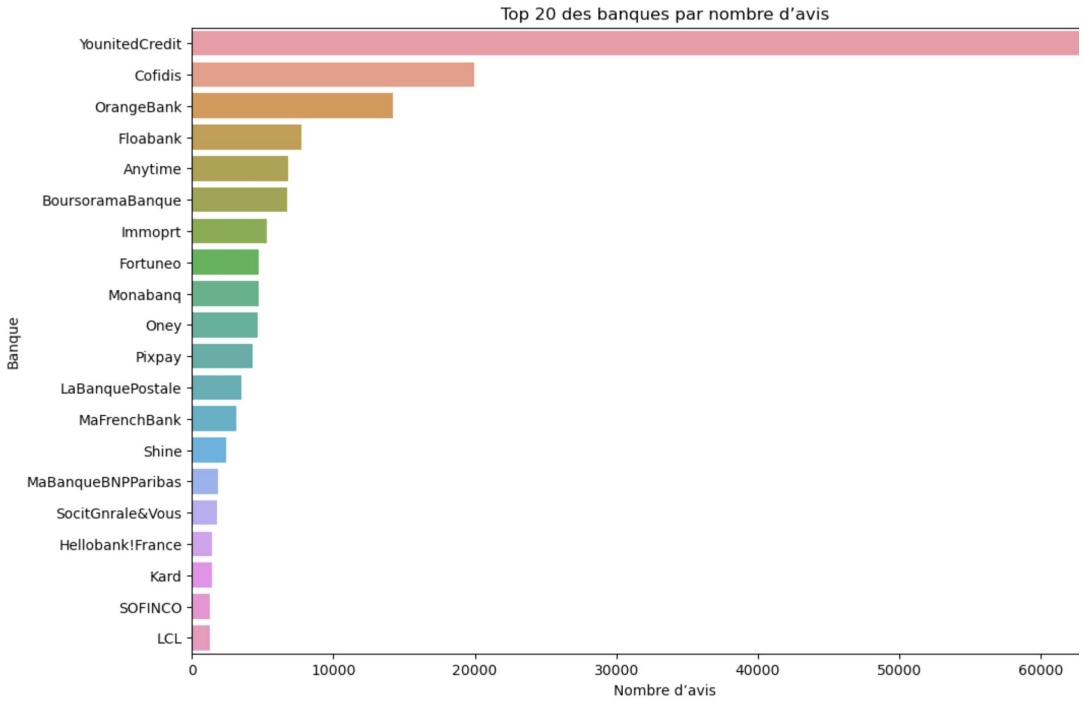
```
Analyse descriptive pour etoiles:
 count    168806.000000
mean          4.241656
std           1.405527
min           1.000000
25%           4.000000
50%           5.000000
75%           5.000000
max           5.000000
Name: etoiles, dtype: float64
```



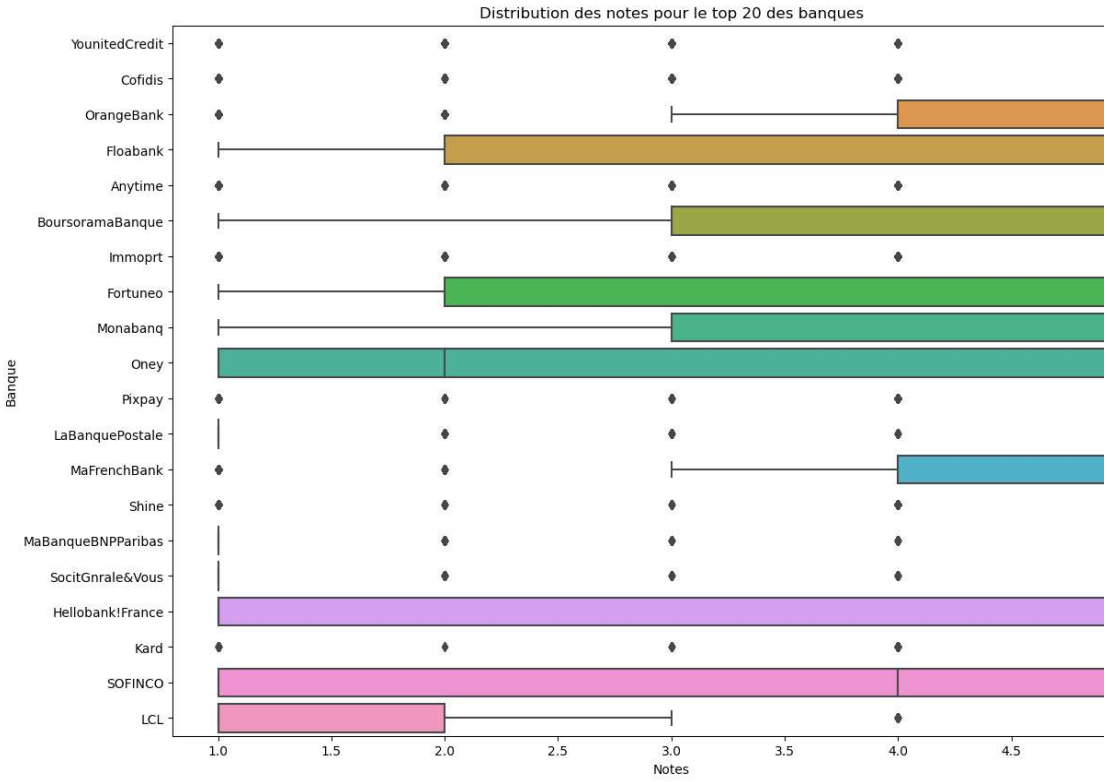
Plus en détail, avec 3010 observations, les deux étoiles sont la modalité la plus minoritaire dans le jeu de données :

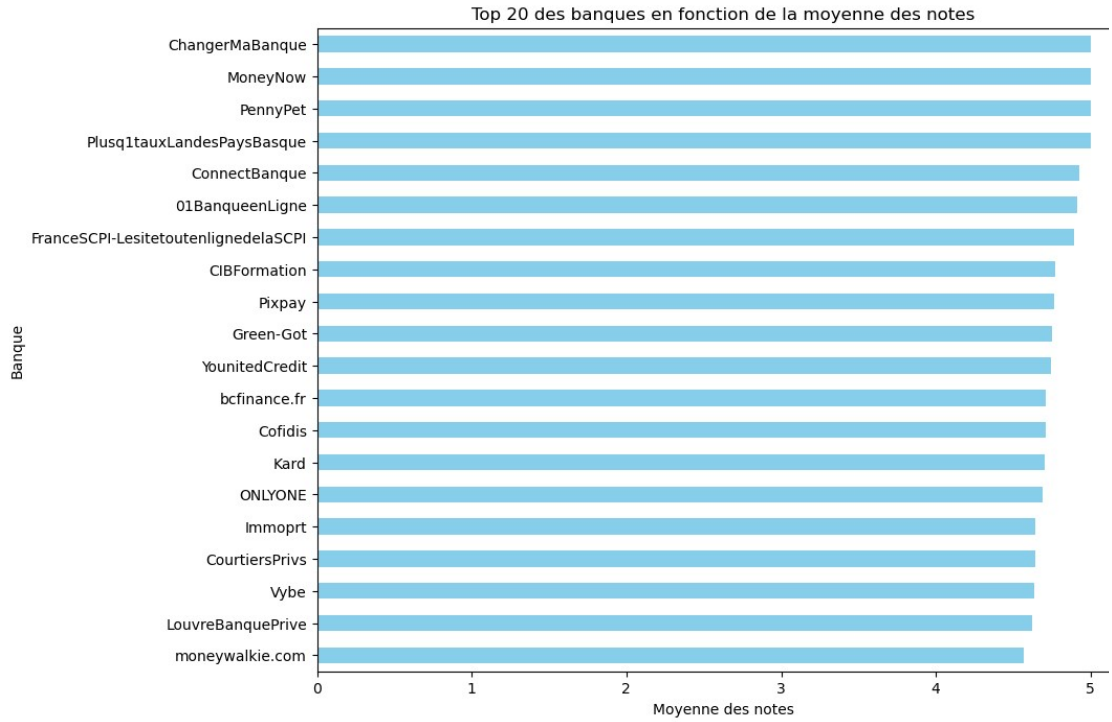
```
etoiles
5    120513
1     23048
4     18763
3      4154
2      3010
Name: count, dtype: int64
```

Le jeu de données contient des avis sur 56 entreprises, YounitedCredit est l'objet du plus grand nombre d'avis, devançant la deuxième banque, Cofidis, par un facteur proche de 3. Nous avons décidé d'afficher uniquement le Top 20 afin de facilier la lecture du graphique.



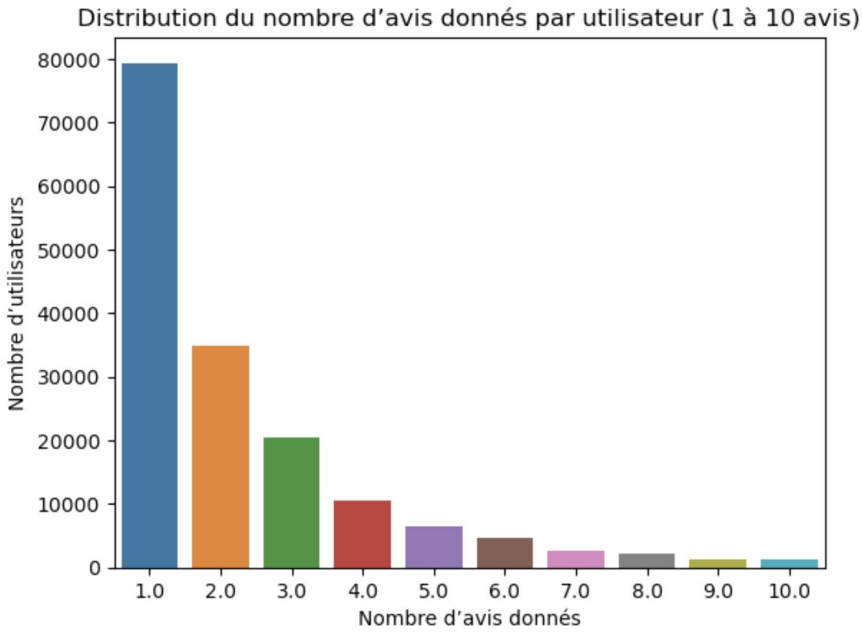
Le graphique suivant nous montre la distribution des notes parmis les banques du Top 20 par nombre d'avis reçus, le suivant montre le Top 20 des Banques par note moyenne.



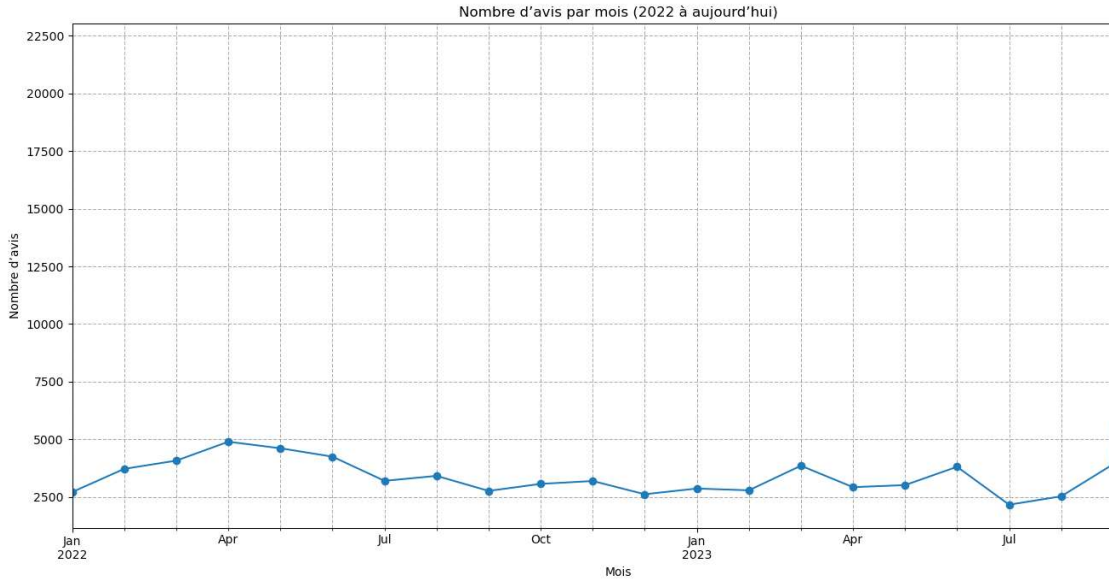


Nous pouvons constater que seulement 5 Banques sont présentes dans les deux classements : Younited, Cofidis, Pixpay, Immoprêt et Kard.

Nous avons également voulu connaître la distribution du nombre d'avis toutes catégories confondues donné par les utilisateurs. La majorité n'a publié qu'un seul avis (80,000), suivi de 2 (35,000 utilisateurs) et 3 avis, au-delà de 4 avis, le nombre d'utilisateurs passe sous les 1000 et décroit rapidement. Le maximum d'avis donnés est de 167, il s'agit probablement d'un bot. Nous devrons retirer cette valeur.

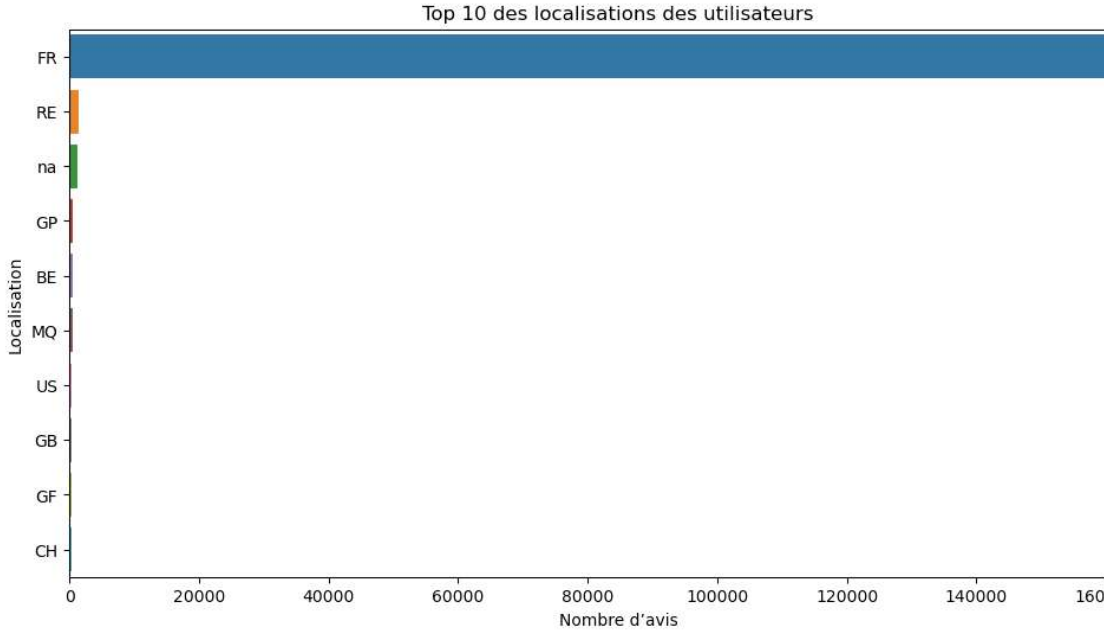


Nous avons également voulu connaitre la distribution des avis dans le temps. De manière assez surprenante, un pic très important arrive sur le dernier mois, celui de la collecte.
A ce stade, nous n'avons pas d'explication claire sur ce pic. Nous envisageons quelques hypothèses : sur le dernier mois des entreprises listées sous d'autres catégorie (assurance) ont été rattachées à la catégorie banque, le nombre d'avis peut également être lié au récent durcissement des conditions de crédit suite à la hausse des taux. De plus, un nombre potentiellement important de nouveaux avis sont effacés chaque mois car ils sont contraires à la charte du site (injures, accusations infondées et/ou bots), il est probable que le nettoyage n'a pas encore été effectuée sur le mois actuel au moment de notre scrapping.

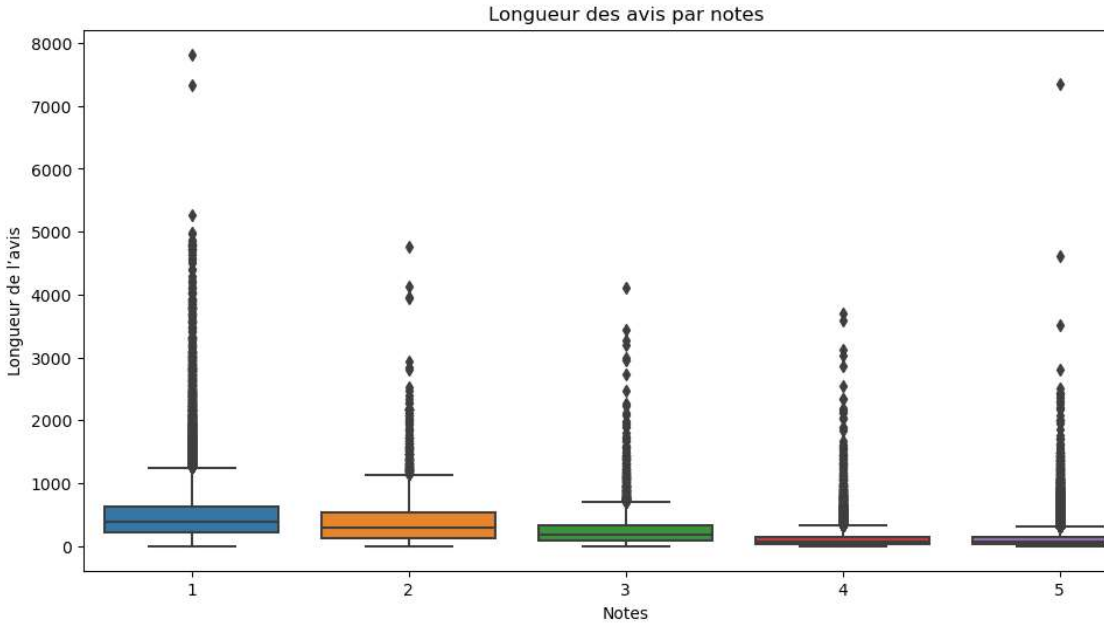


Comme mentionné plus tôt dans la partie nettoyage, nous avons remarqué que la

quasi totalité des avis proviennent de France, c'est pourquoi nous avons décidé de drop cette variable de notre dataframe.



Vu notre problématique, nous pensons utiliser CamemBert pour faire une analyse de sentiments. C'est pourquoi nous avions besoin de voir la longueur des différents avis. A première vue, il semblerait que la longueur de ceux-ci ainsi que la note donnée par les utilisateurs sont corrélés pour les notes extrêmes.



Preprocessing et premiers résultats de modèles

Renforcement du jeu de données

Pour la modélisation nous avons prévu de renforcer le jeu de données en rajoutant certaines variables :

- La longueur des avis au vu de la potentielle relation entre ceux-ci et les notes extrêmes
- Des colonnes pour le jour, le mois et l'année des avis. Ainsi que des variables jour / nuit et semaine / week-end afin d'analyser la temporalité des post et de détecter si des avis ont été publiés en masse (botting).
- Retirer les espaces inutiles ainsi que supprimer les majuscules / caractères spéciaux pour une analyse plus correcte par CamemBert

Nous avons ensuite utilisé le modèle de langage de Deep Learning (Transformers) CamemBert pour établir le sentiment général de l'avis. Dans son analyse de sentiment, CamemBert retourne un label (soit positif, soit négatif) et un score de confiance dans sa labélisation. Pour le renforcement de notre jeu de données, nous combinons les deux, en réétiquetant positif en 1 et négatif en -1, et en multipliant ce résultat par le score de confiance, de manière à avoir des scores entre -1 (très probablement négatif) et 1 (très probablement positif).

Essais de modélisation

Nous avons prévu de faire un benchmark de différents modèles dont SVM, KNN, Random Forest et Camembert pour la classification des étoiles. Baseline : pour 5 modalités, le score d'étiquetage aléatoire serait en moyenne de 0,20.

SVM simple

La prédiction de SVM simple donne un résultat deux fois meilleur qu'un étiquetage aléatoire à 0,40, mais qui reste en deça d'une classification performante.

col_0	1	2	3	4	5
etoiles					
1	475	10	41	61	165
2	354	11	61	109	222
3	208	2	55	188	298
4	43	1	20	441	254

5 100 0 1 104 539

	precision	recall	f1-score	support
1	0.40	0.63	0.49	752
2	0.46	0.01	0.03	757
3	0.31	0.07	0.12	751
4	0.49	0.58	0.53	759
5	0.36	0.72	0.49	744
accuracy			0.40	3763
macro avg	0.40	0.40	0.33	3763
weighted avg	0.40	0.40	0.33	3763

Par la suite nous tenterons de faire une grille pour essayer différents paramètres.

Tests sur CamemBert

Le test de CamemBert sur un échantillon réduit (15500 observations équilibrées entre les modalités – 3100 par étoiles) donne d'emblée de bons résultats, avec un score de 0,7.
Nous pourrons améliorer le score en rajoutant des poids et en faisant tourner le modèle sur l'ensemble du jeu, étant donné que les modèles Transformers sont relativement moins sensibles aux déséquilibres de modalités que les modèles plus classiques de Machine Learning.

Random Forest

Rapport de classification :

Classe 1 (1 étoile) : Précision de 78%, rappel de 62% (F1-score de 69%)
Classe 2 (2 étoiles) : Précision de 100%, rappel de 7% (F1-score de 13%)
Classe 3 (3 étoiles) : Précision de 100%, rappel de 6% (F1-score de 12%)
Classe 4 (4 étoiles) : Précision de 100%, rappel de 2% (F1-score de 3%)
Classe 5 (5 étoiles) : Précision de 79%, rappel de 99% (F1-score de 88%)

Le modèle semble bien performer pour les classes extrêmes (1 et 5 étoiles), mais il a du mal avec les classes intermédiaires, ce qui n'est pas inhabituel dans les distributions déséquilibrées où certaines classes sont beaucoup plus représentées que d'autres. Il a tendance à favoriser la classe majoritaire (ici, 5 étoiles).

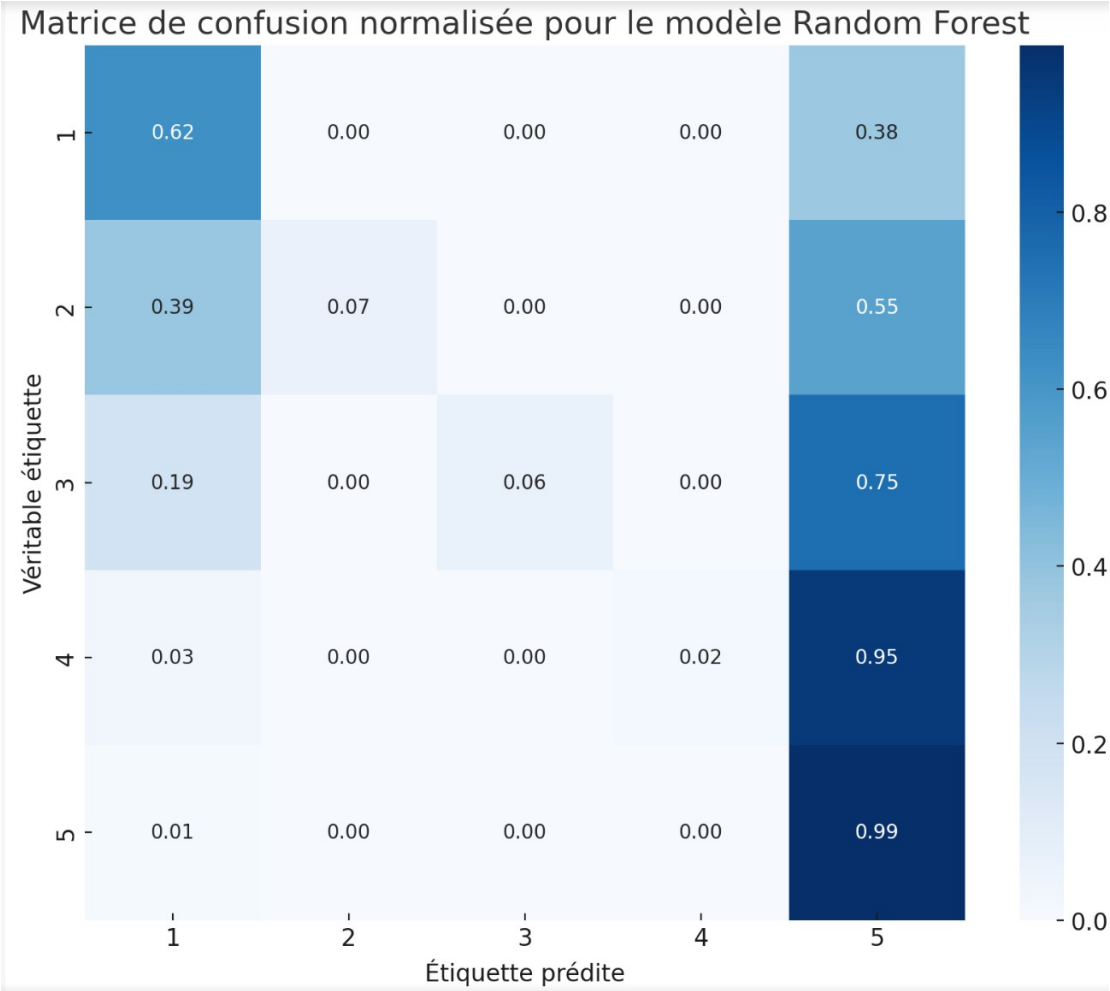
La matrice de confusion normalisée affichée ci-dessous indique les performances du modèle Random Forest sur l'ensemble de test. Les valeurs de la matrice sont normalisées par le nombre d'observations réelles pour chaque classe, ce qui nous permet de voir la proportion des prédictions correctes par rapport au total des cas pour chaque classe réelle.

Dans cette matrice :

Les lignes représentent les classes réelles (le nombre d'étoiles réel donné par les clients).

Les colonnes représentent les classes prédites par le modèle.
Les valeurs sur la diagonale (du haut à gauche au bas à droite) montrent la proportion de prédictions correctes pour chaque classe.
Les autres valeurs indiquent les proportions de classifications incorrectes, où la classe prédite diffère de la classe réelle.

Un modèle parfait aurait 1.0 sur toute la diagonale et 0.0 partout ailleurs. Ici, nous pouvons observer que le modèle prédit relativement bien les avis à 5 étoiles mais performe moins bien pour les autres classes, en particulier les avis à 2, 3 et 4 étoiles, qui sont souvent confondus avec les avis à 5 étoiles. Cela peut indiquer un déséquilibre des classes dans les données d'entraînement, où les avis à 5 étoiles sont probablement plus fréquents, et le modèle a appris à favoriser cette classe.



Data Audit

	Number of lines in the table : *Cleaned : 168806 // Original : 170751*								
# Column	**Name of the Column**	**Variable's type**	**Description**	**Is the variable available before prediction**	**Variable's type**	**Percentage of missing values (in original df, 0 in clean)**	**Categorical / Quantitative**	**Distribution**	**Comments**
		Is the variable a feature or the target ? (Only applicable for supervised learning projects)	*What does this variable represent (from a business perspective ?)*	*Is this variable known before the prediction is made ? (Only applicable for supervised learning projects)*	*int64, float etc...* *If "object", develop.*	*in %*		*For categorical variables with less than 10 categories, list all categories.* *For quantitative variables, detail the distribution (basic descriptive statistics)*	*Free text*
1	user		Le nom de l'utilisateur ayant posté la review	yes	object -> str; il s'agit uniquement d'username	0,34%			
2	etoiles	target	La note donnée avec l'avis	Yes and No -> Goal is automatically guess the note if the user forgot to type it	int64	0,34%	Quantitative	1;2;3;4;5 // mean 4,2 // 25% = 4 already	Recommandation / demande d'avis si clie satisfait ?
3	n_avis		Le nombre d'avis que cet utilisateur a posté	yes	float64	0,73%	Quantitative	1 to 167 // mean 2,8 // std 3,9 // 75% = 3	Avis trop nombreux = bots ?
4	date_avis		La date à laquelle l'avis a été posté	yes	datetime	0,73%	Temporelle		
5	titre_avis	feature	Le titre donné l'avis	yes	object -> str; il s'agit uniquement du titre de l'avis	0,73%	Texte		
6	text_avis	feature	Le texte de l'avis	yes	object -> str; il s'agit uniquement du texte de l'avis	0,73%	Texte		
7	date_experience		La date à laquelle l'utilisateur avait rdv avec la banque	yes	datetime	1,13%	Temporelle		
8	Société		Le nom de la banque	yes	object -> str; il s'agit uniquement du nom de la société	1,13%	Categorical - more than 10 categories		