



ÉCOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTÈMES

PROJET : ML - DATA VISUALIZATION - DEVOPS

APPLICATION D'ANALYSE DES PUBLICATIONS

Élèves :

Rabbah Imrane
Moulay El Hassan Maaly
Zobid Yassine

Encadrants:

Mme. Sanae ELFKIHI
M. Mahmoud EL HAMLAOUI
Mme. MOUNIA ABIK

2 janvier 2025

Remerciements

Nous souhaitons exprimer notre profonde gratitude à nos encadrants, dont le soutien et les conseils ont été essentiels à la réussite de ce projet.

Tout d'abord, nous remercions chaleureusement **Mme. Sanae ELFKIHI**, professeure de Machine Learning, pour son accompagnement attentif et ses explications éclairées sur les concepts d'apprentissage automatique qui ont constitué le cœur de notre projet.

Nous adressons également nos sincères remerciements à **M. Mahmoud EL HAMLAOUI**, professeur en DevOps, pour son expertise et ses recommandations pratiques, notamment dans la mise en œuvre des pipelines CI/CD et l'intégration des technologies Docker.

Enfin, nous remercions vivement **Mme. MOUNIA ABIK**, professeure en Data Visualisation, pour son aide précieuse dans l'analyse et l'interprétation des données, ainsi que pour ses conseils sur l'utilisation d'outils de visualisation avancés.

Ce projet n'aurait pas pu être accompli sans leurs précieuses orientations. Nous leur sommes infiniment reconnaissants pour leur disponibilité, leur encouragement et leur bienveillance tout au long de cette aventure académique.

Résumé

Ce projet porte sur l'analyse des commentaires clients liés aux publications de produits, avec pour objectif principal d'attribuer une note de 1 à 5 à chaque commentaire. En intégrant à la fois le contenu textuel et les interactions sociales (likes et dislikes), nous avons développé un modèle machine learning capable de prédire ces notes de manière précise. Ce système permet également de calculer un score global pour évaluer l'opinion générale sur les publications.

À travers l'utilisation de techniques avancées de machine learning, de visualisation des données, et d'outils DevOps tels que Docker et GitHub Actions, ce projet illustre comment exploiter les données clients pour améliorer la satisfaction utilisateur et soutenir la prise de décision stratégique.

Liste des Abréviations

- **CI/CD** : Continuous Integration / Continuous Deployment (Intégration et Déploiement Continus)
- **ML** : Machine Learning (Apprentissage Automatique)
- **KPI** : Key Performance Indicator (Indicateur Clé de Performance)
- **CSV** : Comma-Separated Values (Valeurs Séparées par des Virgules)
- **TF-IDF** : Term Frequency - Inverse Document Frequency
- **SVM** : Support Vector Machine (Machine à Vecteurs de Support)

Introduction générale

Dans un monde où l'expérience utilisateur est devenue un levier stratégique majeur, analyser les retours clients est essentiel pour comprendre et améliorer les produits et services. Ce projet s'inscrit dans cette optique en développant une solution complète d'analyse des commentaires clients associés à des publications.

L'objectif est de fournir une évaluation détaillée des commentaires en prenant en compte leur contenu textuel et les interactions qu'ils suscitent, telles que les likes et dislikes. Cette approche permet non seulement de prédire des scores individuels, mais aussi de générer un score global reflétant l'opinion générale des utilisateurs.

Grâce à des techniques d'apprentissage automatique et à des outils modernes de visualisation et de déploiement, nous avons conçu une application qui allie performance technique et interactivité. Elle offre des insights exploitables pour les entreprises cherchant à optimiser leurs stratégies marketing et leur engagement client.

Table des matières

Introduction générale	4
1 Présentation du Projet et Analyse des Données	9
1.1 Contexte du Projet	9
1.2 Solutions Proposées	10
1.3 Source de données	10
1.3.1 Champs Disponibles	11
1.4 Prétraitement des Données	11
1.4.1 Nettoyage des Textes	11
1.4.2 Gestion des Valeurs Manquantes	12
1.4.3 Encodage des Variables Textuelles	12
1.4.4 Division des Données	12
1.5 Analyse Exploratoire des Données	13
1.5.1 Distribution des Scores	13
1.5.2 Produits les Plus Commentés avec Nombre Total	13
1.5.3 Vérification des valeurs aberrantes	14
1.5.4 Nombre de Commentaires par Année	15
1.5.5 Évolution du Nombre d'Avis dans le Temps	16
1.5.6 Score Moyen au Fil du Temps	16
1.5.7 Nuage de Mots des Résumés Globaux	17
1.5.8 Treemap des Mots les Plus Fréquents dans les Résumés Globaux	17
1.5.9 Nuage de Mots des Résumés pour Chaque Score	18
1.5.10 Treemap des Mots les Plus Fréquents dans les Résumés pour Chaque Score	19
1.5.11 Matrice de similarité des mots entre les scores	20
1.5.12 Matrice de corrélation	20
2 Réalisation	22
2.1 Outils Utilisés	22
2.1.1 Python	22
2.1.2 Bibliothèques Utilisées	22
2.2 Choix des Modèles	24
2.2.1 Métriques d'Évaluation	24
2.2.2 Régression Logistique	25
2.2.3 Arbre de Décision	25
2.2.4 Support Vector Machine (SVM)	25
2.2.5 Random Forest	25
2.2.6 Modèle Séquentiel Basé sur un Réseau de Neurones Profond	26
2.3 Évaluation des Modèles	26
2.3.1 Régression Logistique	26
2.3.2 Arbre de Décision	27
2.3.3 Support Vector Machine (SVM)	28

2.3.4	Random Forest	28
2.3.5	Voting Classifier	29
2.3.6	Modèle Séquentiel Basé sur un Réseau de Neurones Profond	30
2.3.7	Comparaison des Modèles	31
2.4	Pondération des Scores	31
2.4.1	Calcul du Score Pondéré pour Chaque Commentaire	31
2.4.2	Calcul du Score Global Pondéré (M_p)	32
2.4.3	Exemple de Calcul avec Modification	33
3	Implementation	34
3.1	Développement de la Plateforme avec Streamlit	34
3.1.1	Présentation de Streamlit	34
3.1.2	Avantages de Streamlit	34
3.2	Fonctionnalités de l'Application Streamlit	35
3.2.1	Chargement des Commentaires	35
3.2.2	Résultats	35
3.2.3	Tableau de bord	36
3.3	Containerisation avec Docker	38
3.3.1	Présentation de Docker	38
3.3.2	Image Docker	38
3.3.3	Pourquoi Utiliser Docker pour Streamlit	38
3.3.4	Étapes de Containerisation	39
3.4	Déploiement Continu avec GitHub Actions	40
3.4.1	Configuration du Workflow GitHub Actions	41
3.4.2	Avantages de GitHub Actions pour le Projet	42
	Conclusion Générale	43
	Références	44

Table des figures

1.1	logo Kaggle	10
1.2	Schéma de la Structure du Dataset	11
1.3	Distribution des Scores des Commentaires	13
1.4	Produits les Plus Commentés avec Nombre Total de Commentaires	13
1.5	Vérification des valeurs aberrantes à l'aide de boxplots pour différentes variables.	14
1.6	Boxplot de la Longueur des Résumés des Commentaires	14
1.7	Nombre de Commentaires par Année	15
1.8	Évolution du Nombre d'Avis dans le Temps	16
1.9	Score Moyen des Commentaires au Fil du Temps	16
1.10	Nuage de Mots des Résumés Globaux	17
1.11	Treemap des Mots les Plus Fréquents dans les Résumés Globaux	17
1.12	Nuage de Mots des Résumés pour Chaque Score	18
1.13	Treemap des Mots les Plus Fréquents dans les Résumés pour Chaque Score	19
1.14	Matrice de similarité des mots entre les scores	20
1.15	Matrice de corrélation	20
2.1	Logo de Python	22
2.2	Logo de Scikit-learn	22
2.3	Logos de TensorFlow et Keras	23
2.4	Logos de NLTK et SpaCy	23
2.5	Logos de Matplotlib, Seaborn et Plotly	23
2.6	Matrice de Confusion de la Régression Logistique	27
2.7	Matrice de Confusion de l'Arbre de Décision	27
2.8	Matrice de Confusion du SVM	28
2.9	Matrice de Confusion de la Random Forest	29
2.10	Matrice de Confusion du Voting Classifier	30
2.11	Historique d'Entraînement du Modèle Séquentiel	30
3.1	Logos de Streamlit	34
3.2	Capture d'Écran de l'Option de Chargement des Commentaires	35
3.3	Capture d'Écran de la Section des Résultats	36
3.4	Capture d'Écran des Indicateurs Clés de Performance (KPIs)	36
3.5	Visualisations Interactives du resultat	37
3.6	Résumé des Résultats	38
3.7	Logos de Streamlit et Docker	38
3.8	l'Image Docker	40
3.9	l'application sur le localhost	40
3.10	Logo de GitHub	40

Liste des tableaux

2.1	Rapport de Classification de la Régression Logistique	26
2.2	Rapport de Classification de l'Arbre de Décision	27
2.3	Rapport de Classification du SVM	28
2.4	Rapport de Classification de la Random Forest	28
2.5	Rapport de Classification du Voting Classifier	29
2.6	Comparaison des Performances des Modèles	31

Présentation du Projet et Analyse des Données

Introduction

Cette section présente le contexte et les objectifs du projet, ainsi que la méthodologie adoptée pour atteindre ces objectifs. Nous décrivons le dataset utilisé, les étapes de prétraitement des données et les différentes approches proposées pour analyser les commentaires clients.

1.1 Contexte du Projet

L'objectif principal de ce projet est d'analyser les commentaires clients afin de classifier les sentiments exprimés et d'identifier les tendances récurrentes. En exploitant des techniques de machine learning, il est possible de transformer des données textuelles brutes en informations exploitables pour améliorer les produits et services proposés. Ce projet utilise un dataset contenant plus de 568 000 avis consommateurs sur divers produits Amazon, permettant une analyse à grande échelle et une meilleure généralisation des modèles développés.

De plus, le projet intègre les interactions des utilisateurs, telles que les likes et dislikes, pour affiner l'évaluation des commentaires. Les likes et dislikes influencent le score de chaque commentaire individuel en ajustant sa pondération en fonction de l'engagement qu'il suscite. Par exemple, un commentaire avec de nombreux likes peut voir son score positif renforcé, tandis qu'un commentaire avec de nombreux dislikes peut voir son score négatif amplifié. Cette approche permet de refléter non seulement le contenu textuel des commentaires mais aussi la perception collective des utilisateurs.

Par ailleurs, les likes et dislikes sont également pris en compte dans le calcul du score global pondéré. En intégrant ces interactions au niveau agrégé, le score global offre une mesure plus complète de la satisfaction client. Cela permet d'obtenir une vision synthétique qui combine à la fois les sentiments exprimés dans les commentaires et le degré d'engagement de la communauté, offrant ainsi des insights plus précis pour orienter les décisions stratégiques.

1.2 Solutions Proposées

Développement d'une Application de Calcul et Visualisation du Score Global des Interactions sur les Publications

Pour répondre aux besoins identifiés, une application interactive a été développée. Cette application permet de calculer et de visualiser le score global des interactions sur les publications, facilitant ainsi la prise de décision basée sur les données analysées.

Objectifs de l'Application

- **Calcul du Score Global** : Agréger les différents indicateurs d'interaction (likes, dislikes, commentaires, partages) pour générer un score global reflétant l'engagement des utilisateurs.
- **Visualisation des Données** : Présenter les scores et les tendances sous forme de graphiques interactifs (barres, lignes, nuages de mots) pour une interprétation aisée.
- **Filtrage et Exploration** : Permettre aux utilisateurs de filtrer les données par période, produit ou autres critères pertinents pour une analyse ciblée.

1.3 Source de données

Dans notre projet nous avons utilisé des données issues de la plateforme en ligne **Kaggle**. Kaggle est une source de données très riche qui propose une grande variété de données préparées pour l'analyse, ainsi que des outils et des ressources pour faciliter le travail des data scientists et des développeurs. Nous avons choisi d'utiliser Kaggle comme source de données pour notre projet car il offre un accès facile et rapide à des données de qualité qui peuvent être utilisées pour mettre en place des modèles de prédiction fiables et précis.



FIGURE 1.1 – logo Kaggle

Ce dataset contient plus de 568 000 avis consommateurs sur différents produits Amazon. Bien que ce dataset soit également disponible sur d'autres sites de données, il a été jugé utile et est partagé ici pour son ampleur et sa diversité.

Elle comprend les attributs suivants :

- **Total Records** : 568 454
- **Total Columns** : 10
- **Domain Name** : amazon.com
- **File Extension** : CSV

1.3.1 Champs Disponibles

Les champs disponibles dans ce dataset sont :

- **Id** : Identifiant unique de l'avis.
- **ProductId** : Identifiant du produit concerné par l'avis.
- **UserId** : Identifiant de l'utilisateur ayant rédigé l'avis.
- **ProfileName** : Nom du profil de l'utilisateur.
- **HelpfulnessNumerator** : Nombre de fois où l'avis a été jugé utile.
- **HelpfulnessDenominator** : Nombre total de votes pour l'utilité de l'avis.
- **Score** : Note attribuée au produit (1 à 5).
- **Time** : Timestamp de la publication de l'avis.
- **Summary** : Résumé de l'avis.
- **Text** : Texte complet de l'avis.

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

FIGURE 1.2 – Schéma de la Structure du Dataset

1.4 Prétraitement des Données

Le prétraitement des données est une étape cruciale pour améliorer la qualité des données et optimiser les performances des modèles de machine learning. Voici les étapes principales réalisées :

1.4.1 Nettoyage des Textes

Le nettoyage des textes vise à éliminer les éléments inutiles et à standardiser les données textuelles. Les étapes suivantes ont été effectuées :

- **Conversion en minuscules** : Tous les mots des commentaires ont été convertis en minuscules. Par exemple, "Good" devient "good". Cela permet d'éviter des doublons liés à la casse.
- **Suppression de la ponctuation et des chiffres** : Les caractères non alphabétiques, comme les points (".") et les chiffres ("123"), ont été supprimés. Par exemple, "Product 123 is great!" devient "Product is great".
- **Suppression des stopwords** : Les mots non significatifs comme "the", "is", et "and" ont été éliminés à l'aide de la bibliothèque `nlTK`. Cela réduit le bruit dans les données textuelles.
- **Lemmatisation** : Les mots ont été ramenés à leur forme racine. Par exemple, "running" devient "run", et "better" devient "good". Cela aide à regrouper les mots ayant des significations similaires.

Ces étapes ont permis d'obtenir des données textuelles claires et homogènes, prêtes pour l'analyse.

1.4.2 Gestion des Valeurs Manquantes

Les valeurs manquantes dans les colonnes essentielles ont été traitées comme suit :

- Les enregistrements contenant des champs vides, comme `Text` ou `Score`, ont été supprimés pour garantir des données complètes.
- Les lignes où `HelpfulnessDenominator` était égal à zéro ont également été supprimées. Par exemple, un avis avec "`HelpfulnessNumerator = 0`" et "`HelpfulnessDenominator = 0`" est écarté pour éviter des divisions par zéro.

Cette étape assure une base de données propre, sans biais introduit par des données incomplètes.

1.4.3 Encodage des Variables Textuelles

Les textes nettoyés ont été convertis en représentations numériques en utilisant la méthode **TF-IDF** (**Term Frequency-Inverse Document Frequency**). Les étapes principales incluent :

- **Term Frequency (TF)** : Mesure la fréquence d'apparition d'un mot dans un document. Par exemple, si le mot "great" apparaît 3 fois dans un commentaire contenant 100 mots, alors $TF(\text{great}) = 3/100$.
- **Inverse Document Frequency (IDF)** : Réduit l'importance des mots communs dans tous les documents. Par exemple, un mot rare comme "outstanding" aura un poids plus élevé qu'un mot fréquent comme "product".
- **Vecteur TF-IDF** : Chaque commentaire est transformé en un vecteur où chaque dimension correspond à un mot unique de l'ensemble des commentaires. La valeur pour chaque mot est calculée par TF-IDF.

Cette représentation permet de capturer l'importance des mots dans un contexte global et améliore la performance des modèles.

1.4.4 Division des Données

Pour entraîner et évaluer les modèles, les données ont été divisées comme suit :

- **Train (70%)** : Contient les données utilisées pour l'apprentissage des modèles.
- **Validation (20%)** : Sert à ajuster les hyperparamètres et évaluer les performances intermédiaires.
- **Test (10%)** : Réservé à l'évaluation finale des modèles sur des données inédites.

La division a été effectuée en préservant la répartition des scores (*stratification*) pour éviter un déséquilibre entre les classes.

Les étapes de prétraitement, comprenant le nettoyage des textes, la gestion des valeurs manquantes, l'encodage des textes et la division des données, ont permis de transformer les données brutes en un format structuré et prêt pour l'analyse. Cela a également garanti que les modèles puissent extraire des informations pertinentes pour prédire efficacement les scores des avis.

1.5 Analyse Exploratoire des Données

1.5.1 Distribution des Scores

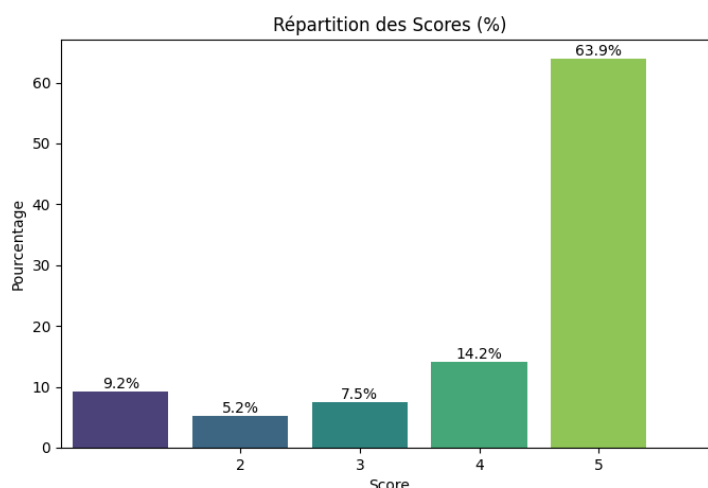


FIGURE 1.3 – Distribution des Scores des Commentaires

Le graphique montre la distribution des scores donnés par les utilisateurs. La majorité des scores (63.9%) sont de 5, indiquant une tendance positive dans les avis. Cette forte concentration des scores élevés peut refléter une satisfaction générale des clients, mais elle pourrait également indiquer un biais dans les évaluations.

1.5.2 Produits les Plus Commentés avec Nombre Total

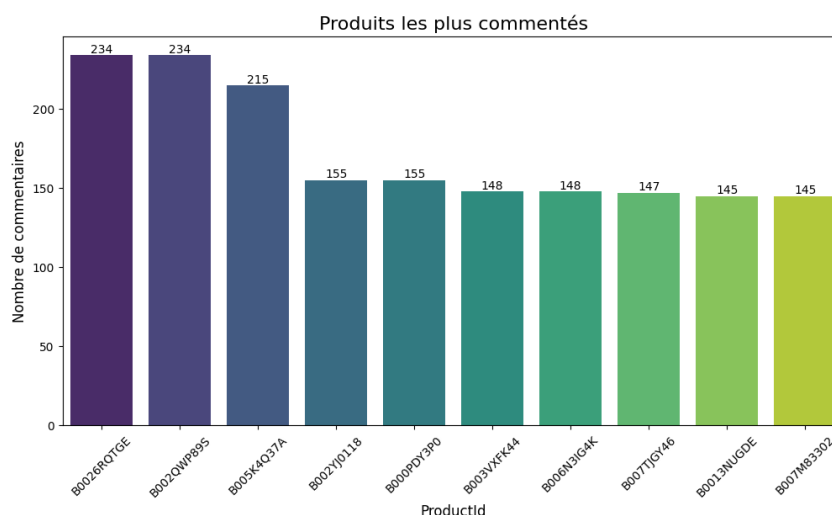


FIGURE 1.4 – Produits les Plus Commentés avec Nombre Total de Commentaires

Ce diagramme en barres présente les produits ayant reçu le plus de commentaires. Les deux premiers produits ont obtenu 234 commentaires chacun, montrant une popularité significative. Ces produits peuvent être ciblés pour une analyse plus approfondie des avis.

1.5.3 Vérification des valeurs aberrantes

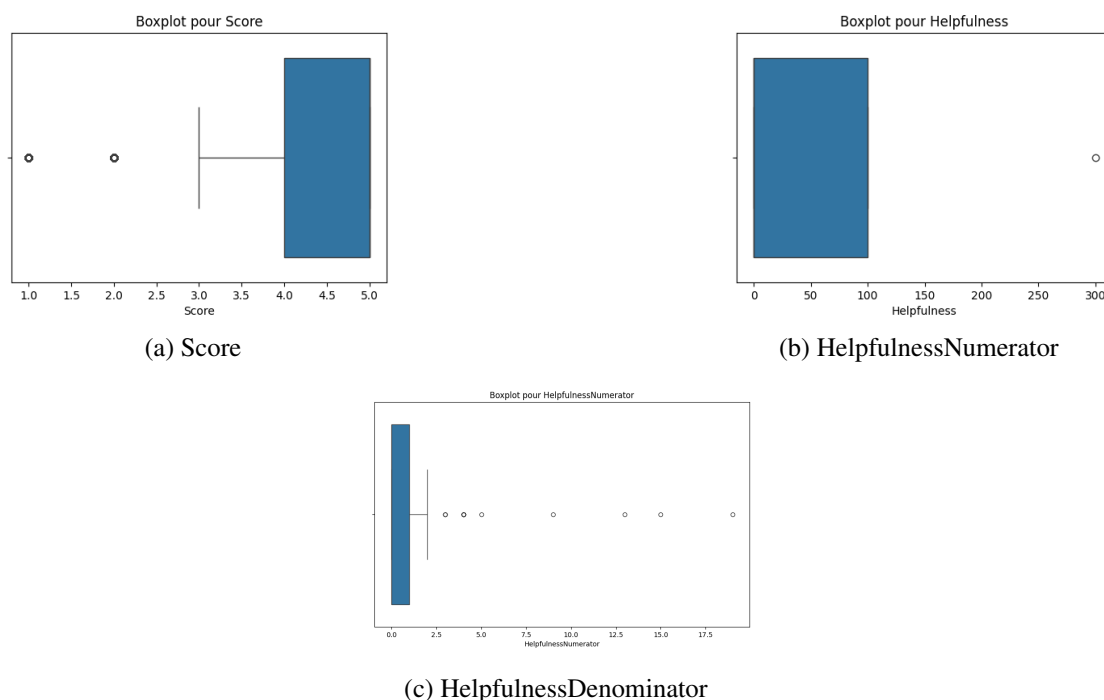


FIGURE 1.5 – Vérification des valeurs aberrantes à l’aide de boxplots pour différentes variables.

Les boxplots ont été utilisés pour détecter les valeurs aberrantes potentielles dans les variables critiques afin d’assurer la qualité des données. Ces valeurs peuvent affecter la précision du modèle si elles ne sont pas traitées correctement.

- **Score (a)** : Quelques valeurs aberrantes ont été détectées, mais la majorité des données sont concentrées autour des scores élevés.
- **HelpfulnessNumerator (b)** : De fortes valeurs aberrantes sont présentes, ce qui pourrait indiquer des cas extrêmes d’interactions sur certains commentaires.
- **HelpfulnessDenominator (c)** : Une distribution similaire avec des valeurs aberrantes significatives.

Les valeurs aberrantes identifiées ont influencé la décision de normaliser ces variables ou de les exclure du modèle dans certains cas critiques.

Boxplot de la Longueur des Résumés

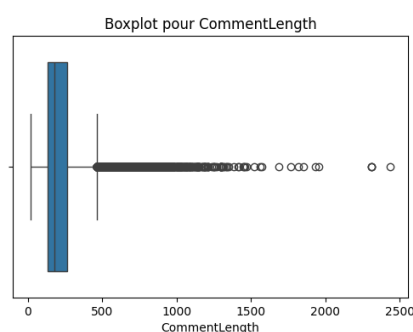


FIGURE 1.6 – Boxplot de la Longueur des Résumés des Commentaires

Ce boxplot est utilisé pour détecter les asymétries dans la longueur des résumés. Cela permet de mieux comprendre la nature des données textuelles.

- La majorité des longueurs des commentaires sont concentrées en dessous de 500 caractères, comme indiqué par la boîte principale.
- Un grand nombre de valeurs aberrantes au-delà de 500 caractères est observé, ce qui peut représenter des commentaires exceptionnellement longs.
- Ces valeurs aberrantes peuvent influencer les résultats des modèles si elles ne sont pas traitées correctement.

Les informations extraites ont motivé l'application d'une transformation ou d'une normalisation pour limiter l'impact des longueurs extrêmes sur les analyses ultérieures.

1.5.4 Nombre de Commentaires par Année

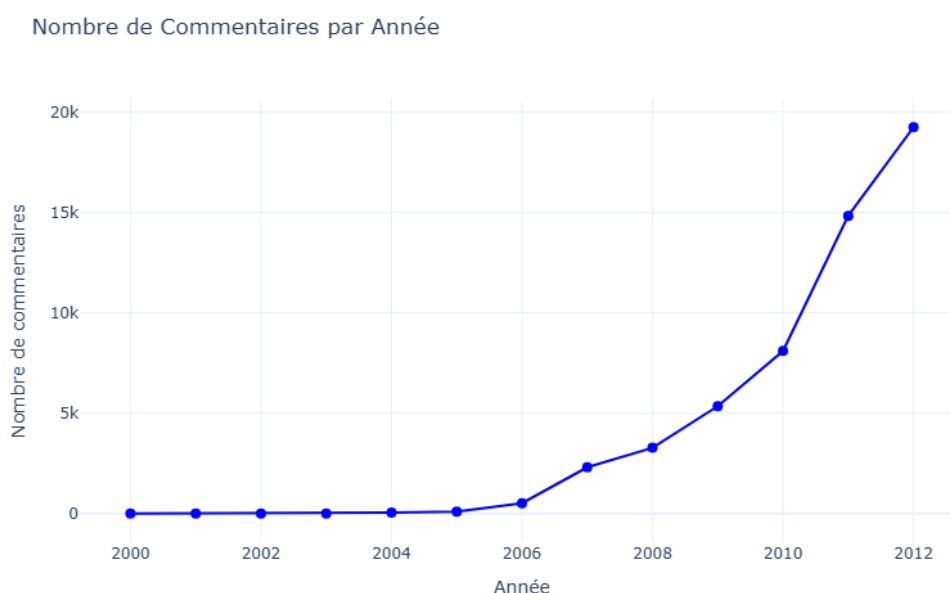


FIGURE 1.7 – Nombre de Commentaires par Année

Ce graphique montre une augmentation exponentielle du nombre de commentaires au fil des ans. Cette croissance reflète l'intérêt croissant pour les avis des produits, mais elle pourrait également être corrélée à l'expansion d'Amazon et à l'adoption des avis par les utilisateurs.

1.5.5 Évolution du Nombre d'Avis dans le Temps

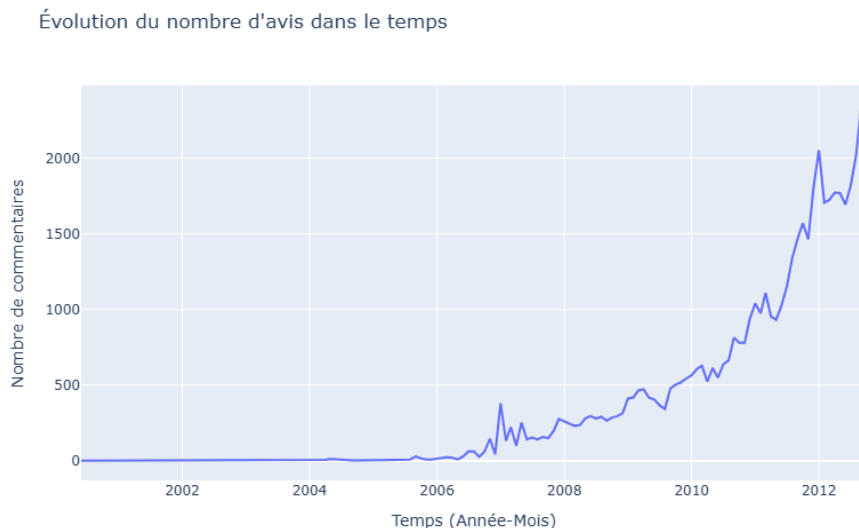


FIGURE 1.8 – Évolution du Nombre d' Avis dans le Temps

ce graphe permet de suivre les tendances temporelles dans les avis publiés. Une forte augmentation du nombre d'avis est observée après 2006, indiquant une adoption massive de la plateforme pour donner des retours.

1.5.6 Score Moyen au Fil du Temps

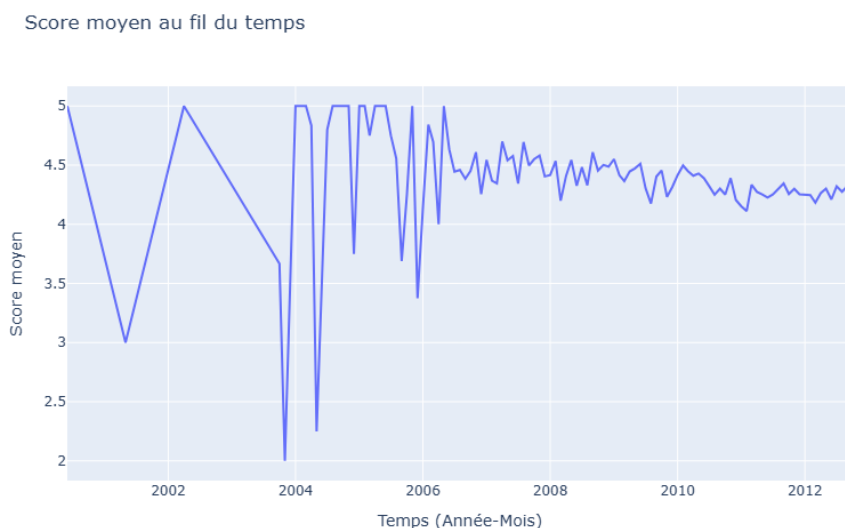


FIGURE 1.9 – Score Moyen des Commentaires au Fil du Temps

Une ligne temporelle représentant l'évolution du score moyen des avis. Une stabilité est observée autour de 4.5 sur une longue période, indiquant une satisfaction relativement constante des clients.

1.5.7 Nuage de Mots des Résumés Globaux



FIGURE 1.10 – Nuage de Mots des Résumés Globaux

Identifier rapidement les mots les plus fréquents dans les résumés des avis.

Les mots *good*, *love*, et *taste* sont les plus fréquents, ce qui reflète une satisfaction générale et des thèmes liés aux préférences gustatives.

1.5.8 Treemap des Mots les Plus Fréquents dans les Résumés Globaux

Treemap des mots les plus fréquents dans les résumés



FIGURE 1.11 – Treemap des Mots les Plus Fréquents dans les Résumés Globaux

Visualiser les mots dominants dans les résumés pour mieux comprendre les thèmes récurrents. Les mots **great**, **love**, et **good** dominent, reflétant une tendance générale positive dans les avis.

1.5.9 Nuage de Mots des Résumés pour Chaque Score



(a) Score 1



(b) Score 2



(c) Score 3



(d) Score 4



(e) Score 5

FIGURE 1.12 – Nuage de Mots des Résumés pour Chaque Score

Les nuages de mots permettent une visualisation qualitative des mots les plus fréquents. Cela aide à identifier les mots-clés associés à des scores spécifiques.

- **Score 1** : Les mots comme "sugar-free" et "bad" dominent, reflétant une expérience négative.
- **Score 2** : Des mots comme "flavor" apparaissent, souvent associés à des critiques mitigées.
- **Score 3** : Les termes "good" et "flavor" indiquent une neutralité dans l'expérience utilisateur.
- **Score 4** : Les mots tels que "oatmeal" et "great" montrent une tendance positive.
- **Score 5** : Les termes "good" et "love" prédominent, indiquant une satisfaction élevée.

Ces informations ont guidé les décisions sur les ajustements des modèles pour mieux refléter ces sentiments.

1.5.10 Treemap des Mots les Plus Fréquents dans les Résumés pour Chaque Score

Treemap des mots les plus fréquents pour le score 1



(a) Score 1

Treemap des mots les plus fréquents pour le score 2



(b) Score 2

Treemap des mots les plus fréquents pour le score 3



(c) Score 3

Treemap des mots les plus fréquents pour le score 4



(d) Score 4

Treemap des mots les plus fréquents pour le score 5



(e) Score 5

FIGURE 1.13 – Treemap des Mots les Plus Fréquents dans les Résumés pour Chaque Score

- **Score 1** : Les mots négatifs comme "bad" dominent.
 - **Score 2** : Des mots mitigés comme "flavor" apparaissent.
 - **Score 3** : Les termes comme "okay" et "good" reflètent une neutralité.
 - **Score 4** : Les mots comme "great" et "good" montrent une satisfaction élevée.
 - **Score 5** : Les mots "great" et "love" prédominent, illustrant une satisfaction exceptionnelle.
- Ces insights ont permis de cibler les améliorations pour les catégories ayant des scores faibles.

1.5.11 Matrice de similarité des mots entre les scores

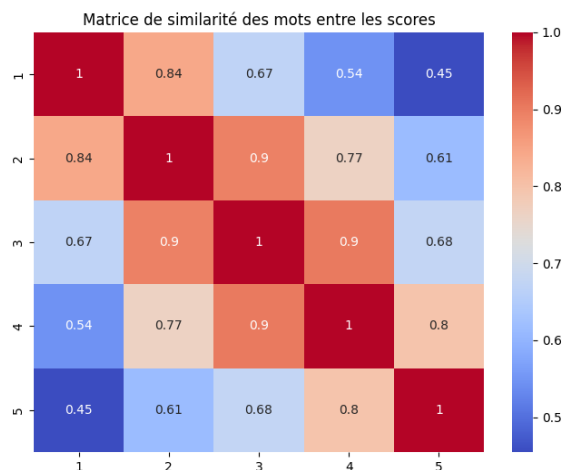


FIGURE 1.14 – Matrice de similarité des mots entre les scores

La matrice de similarité des mots entre les scores permet de comprendre la similarité des mots utilisés dans les avis des différents scores. Une forte similarité est observée entre les scores adjacents, ce qui pourrait indiquer des tendances textuelles communes dans les avis proches.

1.5.12 Matrice de corrélation

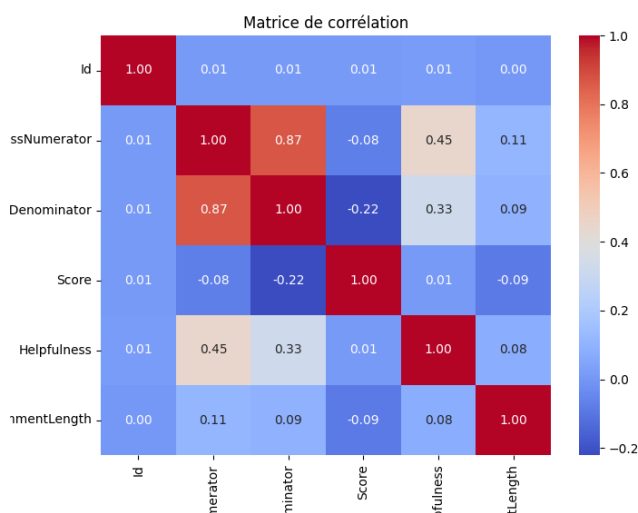


FIGURE 1.15 – Matrice de corrélation

La matrice de corrélation permet d'identifier les relations entre les différentes variables, ce qui est essentiel pour la sélection des caractéristiques.

Dans ce cas, les variables présentent une faible corrélation avec la variable cible Score. Par conséquent, elles ne seront pas considérées comme des variables pertinentes (features) dans le modèle.

Conclusion

Les différentes visualisations exploratoires ont permis de mieux comprendre la distribution et les caractéristiques des données. Les boxplots ont révélé des valeurs aberrantes importantes pour certaines variables, tandis que les nuages de mots et les treemaps ont fourni un aperçu des termes les plus fréquents dans les résumés en fonction des scores. L'évolution temporelle des commentaires a montré une croissance significative, reflétant l'augmentation de l'engagement des utilisateurs au fil des années.

En conclusion, pour la suite de l'analyse, nous avons retenu la variable textuelle des résumés comme feature principale et la variable *Score* comme cible pour nos modèles de machine learning.

Réalisation

2.1 Outils Utilisés

Dans ce projet, plusieurs outils et bibliothèques ont été utilisés pour le développement et l'analyse des données. Chaque outil est présenté avec son logo pour une meilleure visualisation.

2.1.1 Python



FIGURE 2.1 – Logo de Python

Python a été le langage de programmation principal utilisé pour le développement des modèles de machine learning et des visualisations de données. Sa simplicité et sa vaste écosystème de bibliothèques en font un choix idéal pour ce type de projet.

2.1.2 Bibliothèques Utilisées

Scikit-learn



FIGURE 2.2 – Logo de Scikit-learn

Scikit-learn a été utilisé pour implémenter les modèles de machine learning traditionnels tels que la régression logistique, les arbres de décision, le SVM et la random forest. Sa simplicité d'utilisation et sa documentation exhaustive facilitent le développement rapide de modèles robustes.

TensorFlow et Keras



FIGURE 2.3 – Logos de TensorFlow et Keras

TensorFlow, avec son interface haut niveau Keras, a été utilisé pour développer le modèle séquentiel basé sur un réseau de neurones profond. Cette combinaison offre une flexibilité et une puissance accrues pour le traitement des données textuelles complexes.

NLTK et SpaCy

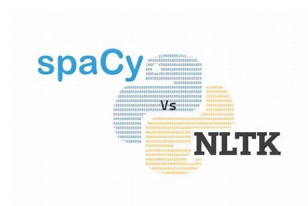


FIGURE 2.4 – Logos de NLTK et SpaCy

NLTK et SpaCy ont été utilisés pour le prétraitement des données textuelles, incluant le nettoyage, la tokenisation, la lemmatisation et l'élimination des stop words. Ces bibliothèques fournissent des outils puissants pour transformer les données textuelles brutes en formats exploitables par les modèles de machine learning.

Matplotlib, Seaborn et Plotly



FIGURE 2.5 – Logos de Matplotlib, Seaborn et Plotly

Matplotlib et Seaborn ont été utilisés pour la création de visualisations statiques telles que les distributions de scores, les boxplots, les nuages de mots et les treemaps. Plotly a été utilisé pour créer des visualisations interactives, facilitant une exploration plus approfondie des données et des résultats des modèles.

2.2 Choix des Modèles

Dans cette section, nous présentons les différents modèles de machine learning sélectionnés pour la classification des sentiments des commentaires clients. Chaque modèle est défini mathématiquement pour illustrer sa structure et son fonctionnement.

2.2.1 Métriques d'Évaluation

Pour évaluer les performances des modèles de classification, plusieurs métriques ont été utilisées. Ces métriques permettent de mesurer la qualité des prédictions effectuées par les modèles et de comparer leur efficacité.

Accuracy (Précision Globale)

L'accuracy est la proportion de prédictions correctes parmi l'ensemble des prédictions effectuées.

$$\text{Accuracy} = \frac{\text{Nombre de Prédictions Correctes}}{\text{Nombre Total de Prédictions}}$$

Précision

La précision mesure la proportion de véritables positifs parmi les prédictions positives.

$$\text{Précision} = \frac{\text{Véritables Positifs}}{\text{Véritables Positifs} + \text{Faux Positifs}}$$

Recall (Rappel)

Le rappel mesure la proportion de véritables positifs correctement identifiés parmi l'ensemble des véritables positifs.

$$\text{Rappel} = \frac{\text{Véritables Positifs}}{\text{Véritables Positifs} + \text{Faux Négatifs}}$$

F1-Score

Le F1-Score est la moyenne harmonique de la précision et du rappel. Il fournit un équilibre entre ces deux métriques.

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Support

Le support représente le nombre réel d'occurrences de chaque classe dans les données de test.

2.2.2 Régression Logistique

La régression logistique est un modèle de classification linéaire utilisé pour prédire la probabilité qu'un échantillon appartienne à une classe spécifique. Pour un problème de classification binaire, la fonction de prédiction est donnée par :

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}}$$

où :

- $P(y = 1|X)$ est la probabilité que l'échantillon appartienne à la classe 1.
- β_0 est l'ordonnée à l'origine.
- $\beta_1, \beta_2, \dots, \beta_n$ sont les coefficients des variables explicatives X_1, X_2, \dots, X_n .

2.2.3 Arbre de Décision

Les arbres de décision sont des modèles non linéaires qui segmentent les données en fonction des caractéristiques les plus discriminantes. Chaque nœud interne de l'arbre représente une condition sur une variable, et chaque branche représente le résultat de cette condition. La structure de l'arbre peut être décrite récursivement :

$$\text{Décision}(X) = \begin{cases} \text{Classe } y_1 & \text{si } X_i \leq \theta \\ \text{Classe } y_2 & \text{sinon} \end{cases}$$

où X_i est une caractéristique particulière et θ est un seuil déterminé lors de la construction de l'arbre.

2.2.4 Support Vector Machine (SVM)

Les SVM sont des modèles de classification qui cherchent à trouver l'hyperplan optimal séparant les différentes classes. L'hyperplan est défini de manière à maximiser la marge entre les classes. Pour un hyperplan défini par $\mathbf{w} \cdot \mathbf{x} + b = 0$, la marge est donnée par :

$$\text{Marge} = \frac{2}{\|\mathbf{w}\|}$$

Le SVM résout le problème d'optimisation suivant :

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

sous les contraintes :

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i$$

où y_i est la classe de l'échantillon \mathbf{x}_i .

2.2.5 Random Forest

Les Random Forests sont des ensembles d'arbres de décision entraînés sur des sous-échantillons des données. Chaque arbre est construit en utilisant un échantillon bootstrap de l'ensemble de données et en sélectionnant aléatoirement un sous-ensemble de caractéristiques à chaque division. La prédiction finale est obtenue par un vote majoritaire (classification) ou une moyenne (régression) des prédictions des arbres individuels.

$$\text{Prédiction} = \text{Mode}\{h_1(X), h_2(X), \dots, h_B(X)\}$$

où $h_b(X)$ est la prédiction de l'arbre b et B est le nombre total d'arbres dans la forêt.

2.2.6 Modèle Séquentiel Basé sur un Réseau de Neurones Profond

Le modèle séquentiel est un réseau de neurones profond constitué de plusieurs couches empilées de manière séquentielle. Chaque couche effectue une transformation linéaire suivie d'une activation non linéaire. La sortie d'une couche est l'entrée de la suivante.

$$\mathbf{h}^{(l)} = \sigma \left(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \right)$$

où :

- $\mathbf{h}^{(l)}$ est la sortie de la couche l .
- $\mathbf{W}^{(l)}$ et $\mathbf{b}^{(l)}$ sont les poids et biais de la couche l .
- σ est la fonction d'activation (par exemple, ReLU, sigmoid).

Le modèle est entraîné en minimisant une fonction de perte, souvent la perte catégorielle croisée pour les problèmes de classification :

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_{ik} \log(\hat{y}_{ik})$$

où y_{ik} est la vérité terrain et \hat{y}_{ik} est la probabilité prédite pour la classe k .

2.3 Évaluation des Modèles

Après l'entraînement des différents modèles, leurs performances ont été évaluées à l'aide de plusieurs métriques. Les résultats sont présentés dans les tableaux et les graphiques ci-dessous pour une comparaison efficace.

2.3.1 Régression Logistique

TABLE 2.1 – Rapport de Classification de la Régression Logistique

Classe	Précision	Rappel	F1-Score	Support
1	0.64	0.65	0.65	10475
2	0.44	0.20	0.27	6005
3	0.44	0.27	0.33	8564
4	0.50	0.23	0.31	16124
5	0.79	0.95	0.86	72523
Accuracy		73%		
Macro Avg	0.56	0.46	0.49	113691
Weighted Avg	0.69	0.73	0.69	113691

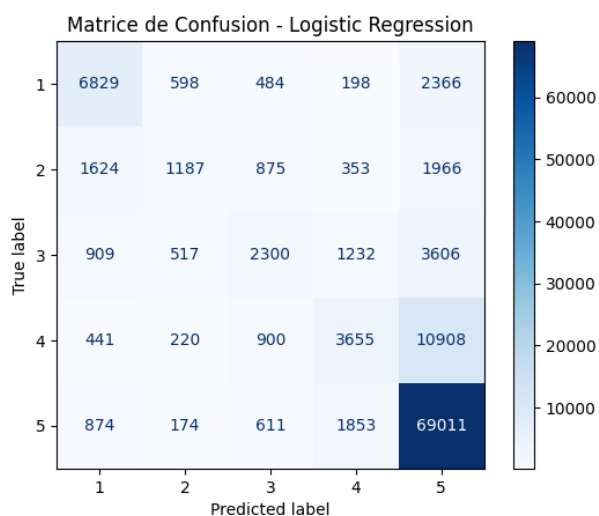


FIGURE 2.6 – Matrice de Confusion de la Régression Logistique

2.3.2 Arbre de Décision

TABLE 2.2 – Rapport de Classification de l'Arbre de Décision

Classe	Précision	Rappel	F1-Score	Support
1	0.62	0.62	0.62	10475
2	0.51	0.45	0.48	6005
3	0.52	0.48	0.50	8564
4	0.54	0.52	0.53	16124
5	0.84	0.86	0.85	72523
Accuracy	74%			
Macro Avg	0.61	0.59	0.60	113691
Weighted Avg	0.74	0.74	0.74	113691

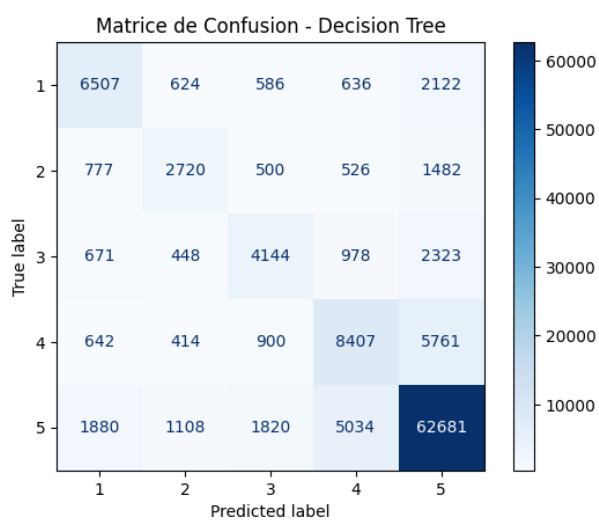


FIGURE 2.7 – Matrice de Confusion de l'Arbre de Décision

2.3.3 Support Vector Machine (SVM)

TABLE 2.3 – Rapport de Classification du SVM

Classe	Précision	Rappel	F1-Score	Support
1	0.63	0.68	0.65	10475
2	0.45	0.20	0.24	6005
3	0.42	0.27	0.33	8564
4	0.50	0.25	0.31	16124
5	0.78	0.92	0.86	72523
Accuracy	74%			
Macro Avg	0.57	0.45	0.49	113691
Weighted Avg	0.70	0.74	0.69	113691

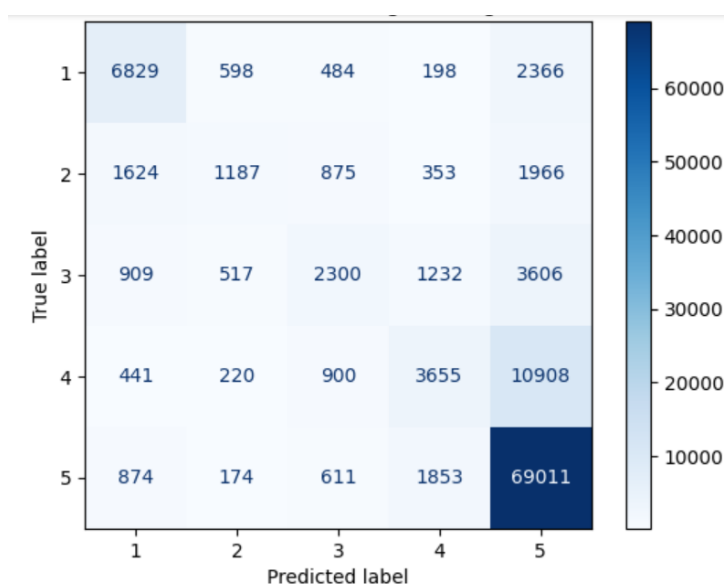


FIGURE 2.8 – Matrice de Confusion du SVM

2.3.4 Random Forest

TABLE 2.4 – Rapport de Classification de la Random Forest

Classe	Précision	Rappel	F1-Score	Support
1	0.83	0.63	0.72	10475
2	0.97	0.39	0.55	6005
3	0.93	0.42	0.58	8564
4	0.92	0.43	0.59	16124
5	0.78	0.99	0.88	72523
Accuracy	81%			
Macro Avg	0.89	0.57	0.66	113691
Weighted Avg	0.83	0.81	0.78	113691

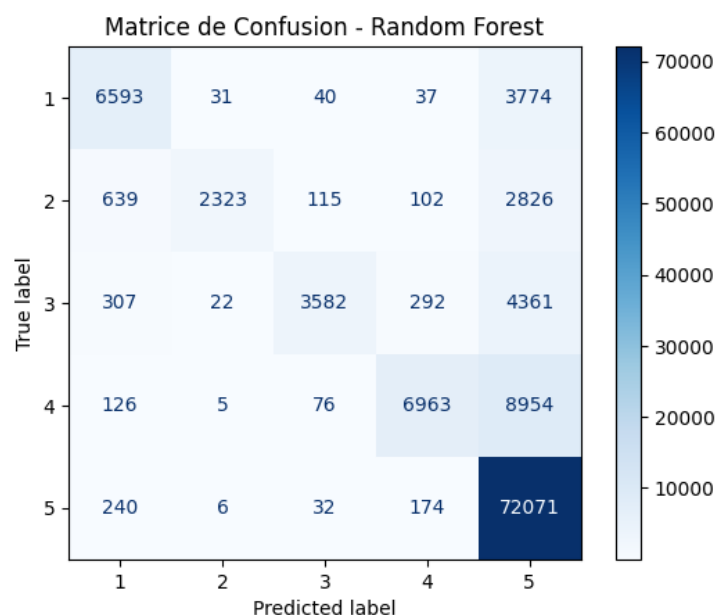


FIGURE 2.9 – Matrice de Confusion de la Random Forest

2.3.5 Voting Classifier

Description

Le Voting Classifier combine plusieurs modèles de machine learning pour améliorer les performances globales en agrégeant leurs prédictions. Dans ce projet, nous avons utilisé les modèles suivants :

- Régression Logistique
- Arbre de Décision
- Support Vector Machine (SVM)
- Random Forest

Cette approche permet de tirer parti des forces de chaque modèle individuel pour obtenir une performance globale supérieure.

Rapport de Classification

TABLE 2.5 – Rapport de Classification du Voting Classifier

Classe	Précision	Rappel	F1-Score	Support
1	0.74	0.63	0.68	962
2	0.69	0.29	0.41	434
3	0.67	0.31	0.42	503
4	0.75	0.29	0.42	1251
5	0.83	0.98	0.90	7608
Accuracy	81%			
Macro Avg	0.74	0.50	0.57	10758
Weighted Avg	0.80	0.81	0.78	10758

Matrice de Confusion

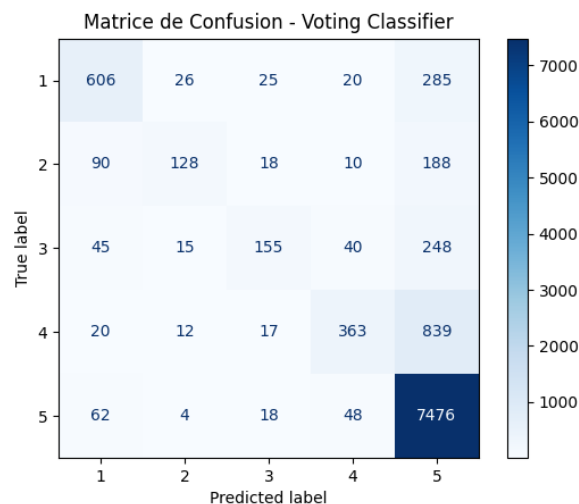


FIGURE 2.10 – Matrice de Confusion du Voting Classifier

Le modèle a une bonne précision pour les scores élevés (notamment le score 5), mais une précision moindre pour les scores intermédiaires (par exemple, les scores 2 et 3). Cela indique une difficulté à différencier les classes voisines.

2.3.6 Modèle Séquentiel Basé sur un Réseau de Neurones Profond

Rapport d'Entraînement

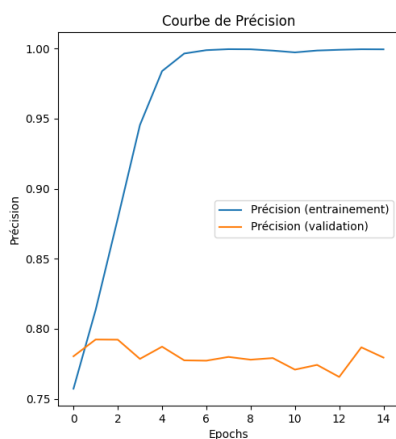


FIGURE 2.11 – Historique d'Entraînement du Modèle Séquentiel

La précision d'entraînement atteint presque 100%, mais la précision de validation stagne autour de 78%. Cela indique un surapprentissage potentiel du modèle.

Epoch 15/15

1076/1076 15s 14ms/step – accuracy : 0.9995 – loss : 0.0014 – val_accuracy : 0.7795 – val_loss : 2.4741

2.3.7 Comparaison des Modèles

Tableau Comparatif

Le tableau 2.6 compare les performances des différents modèles utilisés dans ce projet.

TABLE 2.6 – Comparaison des Performances des Modèles

Modèle	Accuracy	Précision	Rappel
Régression Logistique	73%	69%	73%
Arbre de Décision	74%	74%	74%
SVM	74%	70%	69%
Random Forest	81%	78%	81%
Voting Classifier	81%	80%	78%
Modèle Séquentiel	77.95%	74%	73%

Le Voting Classifier a démontré des performances supérieures par rapport aux modèles individuels, atteignant une précision de 81%. Cette amélioration est attribuée à l'agrégation des forces de chaque modèle, permettant une meilleure généralisation et une robustesse accrue face aux variations des données. Le modèle séquentiel basé sur un réseau de neurones profond a également montré de bonnes performances, bien qu'inférieures au Voting Classifier dans ce contexte spécifique. Toutefois, le modèle séquentiel a été considéré pour la plateforme en raison de sa capacité à capturer des relations non linéaires et complexes, offrant une meilleure flexibilité pour des analyses futures.

2.4 Pondération des Scores

Afin de refléter non seulement le contenu textuel des commentaires mais aussi l'engagement des utilisateurs (likes et dislikes), une pondération des scores a été mise en place. Cette pondération permet d'ajuster les scores des commentaires individuels et de calculer un score global pondéré.

2.4.1 Calcul du Score Pondéré pour Chaque Commentaire

Le score pondéré (score_pondéré) est calculé en fonction du score initial attribué par le modèle de prédiction, ainsi que du nombre de likes et de dislikes associés à chaque commentaire. La formule est différente selon que le score initial est positif ou négatif.

Formule Unifiée pour le Score Pondéré

$$\text{score_pondéré} = \begin{cases} \text{score_initial} \times (1 + \alpha \times \text{likes} - \beta \times \text{dislikes}) & \text{si } \text{score_initial} \geq 3, \\ \text{score_initial} \times (1 + \alpha \times \text{dislikes} - \beta \times \text{likes}) & \text{si } \text{score_initial} \leq 2. \end{cases}$$

- α : Coefficient pondérant l'impact des likes.
- β : Coefficient pondérant l'impact des dislikes.

Borne et Arrondissement du Score Final

Après le calcul du score pondéré, nous assurons que le score final reste dans une plage acceptable entre 1 et 5, et nous arrondissons par troncature.

$$\text{score_final} = \lfloor \max(1, \min(\text{score_pondéré}, 5)) \rfloor$$

- $\max(1, \dots)$ garantit que le score final ne descend pas en dessous de 1.
- $\min(\dots, 5)$ garantit que le score final ne dépasse pas 5.
- $\lfloor \cdot \rfloor$ représente l'arrondissement par troncature.

2.4.2 Calcul du Score Global Pondéré (M_p)

Le score global pondéré (M_p) est une mesure agrégée de la satisfaction globale des clients, calculée en tenant compte de la distribution des scores finaux des commentaires et de l'engagement de la publication.

Facteur de Normalisation

$$\delta = 5$$

- δ est le facteur de normalisation, représentant le score maximal possible.

Moyenne Globale (μ)

$$\mu = \frac{1}{N} \sum_{i=1}^N C_i$$

- C_i : Score final du $i^{\text{ème}}$ commentaire.
- N : Nombre total de commentaires.
- μ : Moyenne des scores finaux.

Calcul des Poids (w_i)

$$w_i = 1 - \frac{|C_i - \mu|}{\delta}$$

- w_i : Poids attribué au $i^{\text{ème}}$ commentaire.
- Cette formule attribue des poids plus élevés aux scores proches de la moyenne globale, reflétant leur importance dans le calcul du score global pondéré.
- Les poids sont ensuite bornés pour s'assurer qu'ils restent positifs :

$$w_i = \max(0, w_i)$$

Calcul du Score Global Pondéré

$$M_p = \frac{\sum_{i=1}^N C_i \times w_i + \alpha \times \text{Total Likes Publication} - \beta \times \text{Total Dislikes Publication}}{\sum_{i=1}^N w_i + \alpha + \beta}$$

- M_p : Score global pondéré.
- Cette formule agrège les scores finaux pondérés, en intégrant l'engagement de la publication (likes et dislikes) pour fournir une mesure synthétique de la satisfaction globale des clients.

Borne et Arrondissement du Score Global

$$M_p = \lfloor \max(1, \min(M_p, 5)) \rfloor$$

- Le score global pondéré est d'abord borné entre 1 et 5.
- Ensuite, il est arrondi au nombre entier le plus proche pour simplifier l'interprétation.

2.4.3 Exemple de Calcul avec Modification

Données :

- Commentaire : "Frustrating shopping experience, unreliable website."
- Likes (likes) : 2
- Dislikes (dislikes) : 20
- Score Initial (Score) : 3

Calcul du Score Pondéré :

$$\text{score_pondéré} = 3 \times (1 + 0.04 \times 2 - 0.01 \times 20) = 3 \times (1 + 0.08 - 0.2) = 3 \times 0.88 = 2.64$$

$$\text{score_final} = \lfloor \max(1, \min(2.64, 5)) \rfloor = \lfloor 2.64 \rfloor = 2$$

Calcul du Poids :

$$w_i = 1 - \frac{|2 - 3|}{5} = 1 - \frac{1}{5} = 0.8$$

Calcul du Score Global Pondéré (M_p) : $M_p = \frac{(2 \times 0.8) + 0.04 \times 100 - 0.01 \times 50}{0.8 + 0.04 + 0.01} = \frac{1.6 + 4 - 0.5}{0.85} = \frac{5.1}{0.85} \approx 6$

$$M_p = \lfloor \max(1, \min(6, 5)) \rfloor = \lfloor 5 \rfloor = 5$$

Conclusion

Cette section a détaillé les méthodes d'évaluation des modèles utilisés dans ce projet, ainsi que la pondération des scores pour refléter à la fois le contenu des commentaires et l'engagement des utilisateurs. Les résultats montrent que le Voting Classifier offre des performances supérieures, et le modèle séquentiel a été retenu pour sa capacité à capturer des relations complexes dans les données textuelles.

Implementation

Introduction

Ce chapitre décrit la mise en œuvre pratique du projet, en se concentrant sur le développement de la plateforme interactive et la containerisation avec Docker. Nous présentons les outils et technologies utilisés, les étapes de développement, et incluons des captures d'écran illustrant l'application et l'image Docker.

3.1 Développement de la Plateforme avec Streamlit

3.1.1 Présentation de Streamlit

Streamlit est une plateforme open-source permettant de créer rapidement des applications web interactives pour le machine learning et la science des données. Elle simplifie le processus de déploiement des modèles de machine learning en offrant une interface utilisateur intuitive sans nécessiter de compétences approfondies en développement web.



FIGURE 3.1 – Logos de Streamlit

3.1.2 Avantages de Streamlit

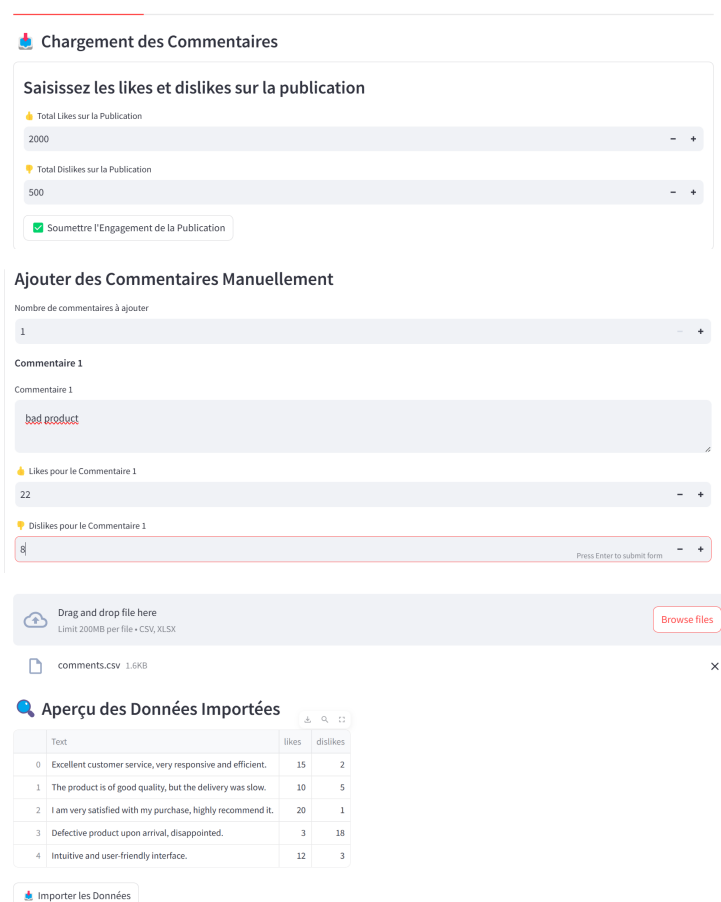
- **Simplicité** : Facile à apprendre et à utiliser avec une syntaxe Python simple.
- **Interactivité** : Permet de créer des interfaces utilisateur interactives pour visualiser et explorer les données.
- **Rapidité** : Déploiement rapide des applications, facilitant le prototypage et le partage des résultats.

3.2 Fonctionnalités de l'Application Streamlit

L'application Streamlit développée pour ce projet inclut plusieurs fonctionnalités clés permettant une interaction efficace avec les données de commentaires clients. Chaque fonctionnalité est décrite ci-dessous avec une capture d'écran illustrant son interface et son fonctionnement.

3.2.1 Chargement des Commentaires

Cette fonctionnalité permet aux utilisateurs d'ajouter le nombre de likes et de dislikes sur la publication, ainsi que d'ajouter des commentaires manuellement ou d'importer des commentaires à partir de fichiers au format CSV ou XLSX.



Chargement des Commentaires

Saisissez les likes et dislikes sur la publication

Total Likes sur la Publication: 2000

Total Dislikes sur la Publication: 500

☒ Soumettre l'Engagement de la Publication

Ajouter des Commentaires Manuellement

Nombre de commentaires à ajouter: 1

Commentaire 1: bad product

Likes pour le Commentaire 1: 22

Dislikes pour le Commentaire 1: 6

Drag and drop file here
Limit 200MB per file • CSV, XLSX

comments.csv 1.6KB

Aperçu des Données Importées

	Text	likes	dislikes
0	Excellent customer service, very responsive and efficient.	15	2
1	The product is of good quality, but the delivery was slow.	10	5
2	I am very satisfied with my purchase, highly recommend it.	20	1
3	Defective product upon arrival, disappointed.	3	18
4	Intuitive and user-friendly interface.	12	3

Importer les Données

FIGURE 3.2 – Capture d'Écran de l'Option de Chargement des Commentaires

3.2.2 Résultats

La section des résultats affiche les résultats pour chaque commentaire, incluant le score initial attribué par le modèle et le score final après pondération. Cela permet de visualiser l'impact des likes et dislikes sur la satisfaction globale des clients.

Détails des Commentaires Analyés

	Commentaire	Likes	Dislikes	Score Initial	Score Final
0	Excellent customer service, very responsive and efficient.	15	2	5	5
1	The product is of good quality, but the delivery was slow.	10	5	4	5
2	I am very satisfied with my purchase, highly recommend it.	20	1	5	5
3	Defective product upon arrival, disappointed.	3	18	1	1
4	Intuitive and user-friendly interface.	12	3	1	1
5	Lacking important features, needs improvements.	5	10	3	3
6	Fast delivery and neat packaging, perfect!	18	0	5	5
7	Technical support failed to resolve my issue.	4	15	1	1
8	Good value for money, satisfied with my investment.	14	4	5	5
9	Frustrating shopping experience, unreliable website.	2	20	3	2

FIGURE 3.3 – Capture d’Écran de la Section des Résultats

3.2.3 Tableau de bord

Indicateurs Clés de Performance (KPIs)

Les KPIs fournissent un aperçu des mesures essentielles liées aux commentaires et à l’engagement des utilisateurs. Ces indicateurs incluent le score global ajusté (M_p), le nombre total de likes et dislikes sur les commentaires, ainsi que sur la publication elle-même.

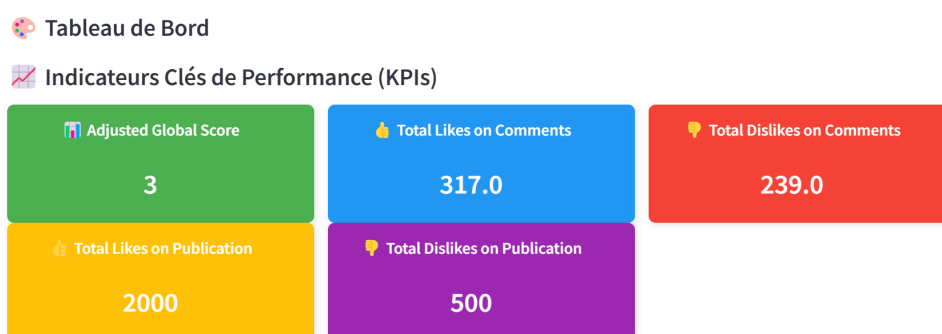


FIGURE 3.4 – Capture d’Écran des Indicateurs Clés de Performance (KPIs)

Visualisations Interactives

Cette fonctionnalité utilise Plotly pour générer des visualisations interactives, telles que des nuages de mots (Word Clouds) des commentaires, permettant une exploration approfondie des données textuelles.

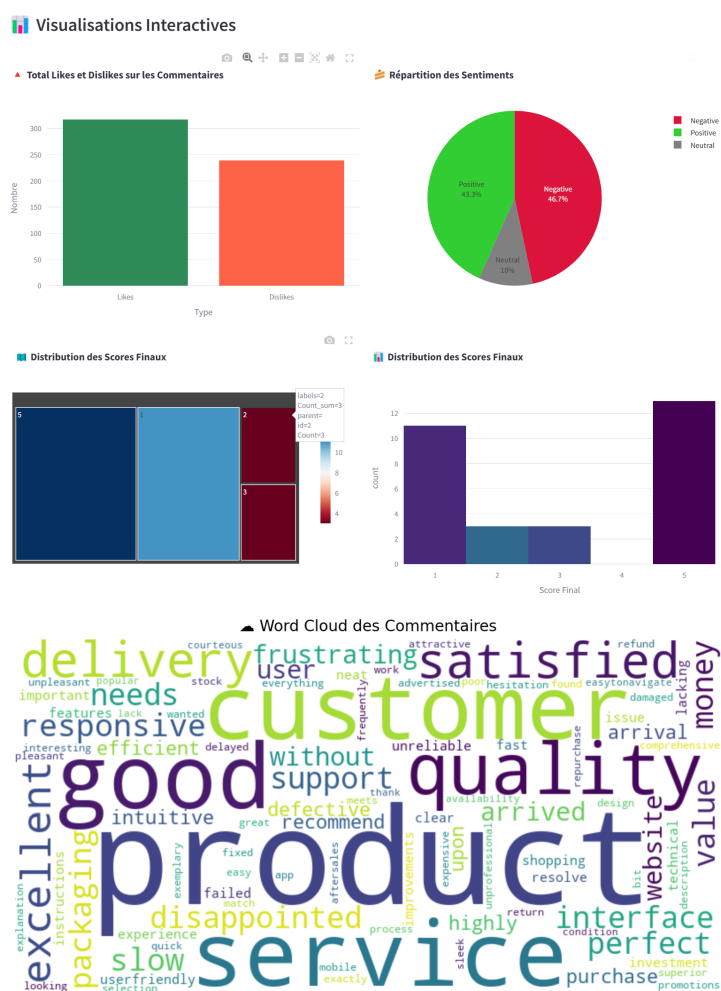


FIGURE 3.5 – Visualisations Interactives du resultat

Générer et Télécharger le Rapport PDF

Les utilisateurs peuvent générer un rapport PDF détaillant les résultats de l'analyse des commentaires. Cette fonctionnalité permet également de télécharger le rapport directement depuis l'application.

Résumé des Résultats

Cette section présente un résumé concis des résultats obtenus, incluant le score global ajusté et une recommandation basée sur l'analyse des données.

Recommendation

En fonction du score global ajusté, l'application fournit des recommandations spécifiques. Par exemple, un score bas peut déclencher une recommandation indiquant la nécessité d'améliorations urgentes pour augmenter la satisfaction des clients.

Nettoyer les Fichiers Temporaires

Cette fonctionnalité permet de nettoyer les fichiers temporaires générés lors du traitement des données, assurant ainsi une gestion efficace de l'espace de stockage et le bon fonctionnement de l'application.


Générer et Télécharger le Rapport PDF ↗


 Générer le Rapport PDF


Résumé des Résultats

 Adjusted Global Score : 3

 **Recommandation :** ⚠ Améliorations modérées nécessaires.


Nettoyer les Fichiers Temporaires

 Nettoyer les Fichiers Temporaires

FIGURE 3.6 – Résumé des Résultats

3.3 Containerisation avec Docker

3.3.1 Présentation de Docker

Docker est une plateforme de containerisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et portables. La containerisation assure que l'application fonctionne de manière cohérente sur différents environnements, facilitant ainsi le déploiement et la scalabilité.



FIGURE 3.7 – Logos de Streamlit et Docker

3.3.2 Image Docker

L'image Docker est une entité légère et autonome qui contient tout le nécessaire pour exécuter l'application Streamlit, y compris le code, les dépendances et les configurations. En utilisant une image Docker, nous assurons une portabilité maximale et une cohérence des environnements de déploiement, facilitant ainsi la distribution et l'exécution de l'application sur différents systèmes.

3.3.3 Pourquoi Utiliser Docker pour Streamlit

- **Portabilité** : Les conteneurs Docker peuvent être exécutés sur n'importe quelle machine disposant de Docker installé.
- **Isolation** : Chaque conteneur fonctionne de manière isolée, évitant les conflits de dépendances.

- **Facilité de Déploiement** : Simplifie le déploiement de l'application Streamlit sur des serveurs ou des services cloud.

3.3.4 Étapes de Containerisation

1. Créer un Fichier Dockerfile

Le fichier 'Dockerfile' définit l'environnement nécessaire pour exécuter l'application.

```

1  # Utiliser une image Python légère comme base
2  FROM python:3.9-slim
3
4  # Définir le répertoire de travail dans le conteneur
5  WORKDIR /app
6
7  # Copier le fichier principal de l'application dans le conteneur
8  COPY app.py app.py
9  COPY fonts/ /app/fonts
10 COPY model_reviews.keras /app/model_reviews.keras
11 COPY tfidf_vectorizer.pkl /app/tfidf_vectorizer.pkl
12
13 # Installer les dépendances système nécessaires
14 RUN apt-get update && apt-get install -y \
15     build-essential \
16     libgl2.0-0 \
17     libsm6 \
18     libxrender1 \
19     libxext6 \
20     && apt-get clean
21
22 # Installer les bibliothèques Python nécessaires
23 RUN pip install --no-cache-dir \
24     streamlit \
25     tensorflow \
26     pandas \
27     numpy \
28     scikit-learn \
29     nltk \
30     wordcloud \
31     matplotlib \
32     plotly \
33     fpdf \
34     kaleido
35
36 # Télécharger les ressources NLTK nécessaires (stopwords)
37 RUN python -m nltk.downloader stopwords
38
39 # Exposer le port par défaut de Streamlit
40 EXPOSE 8501
41
42 # Commande pour exécuter l'application Streamlit
43 CMD ["streamlit", "run", "app.py", "--server.enableCORS=false", "--server.port=8501",
    ↪  "--server.address=0.0.0.0"]

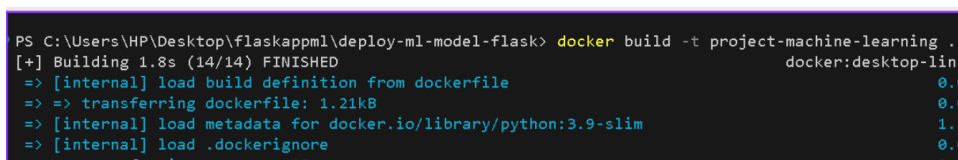
```


2. Construire l'Image Docker

Utilisez la commande suivante pour construire l'image Docker :

```
1 docker build -t project-machine-learning .
```

Listing 3.1 – Commande pour Construire l'Image Docker



```
PS C:\Users\HP\Desktop\flaskappml\deploy-ml-model-flask> docker build -t project-machine-learning .
[+] Building 1.8s (14/14) FINISHED                                docker:desktop-linu
=> [internal] load build definition from dockerfile              0.0
=> => transferring dockerfile: 1.21kB                            0.0
=> [internal] load metadata for docker.io/library/python:3.9-slim 1.5
=> [internal] load .dockerignore                                  0.0
=> => transferring context: 2B                                     0.0
```

FIGURE 3.8 – l'Image Docker

3. Exécuter le Conteneur Docker

Lancez le conteneur en utilisant la commande suivante :

```
1 docker run -d -p 8501:8501 mon_app_streamlit
```

Listing 3.2 – Commande pour Exécuter le Conteneur Docker

4. Accéder à l'Application

Ouvrez un navigateur web et naviguez vers <http://localhost:8501> pour voir l'application Streamlit en cours d'exécution.

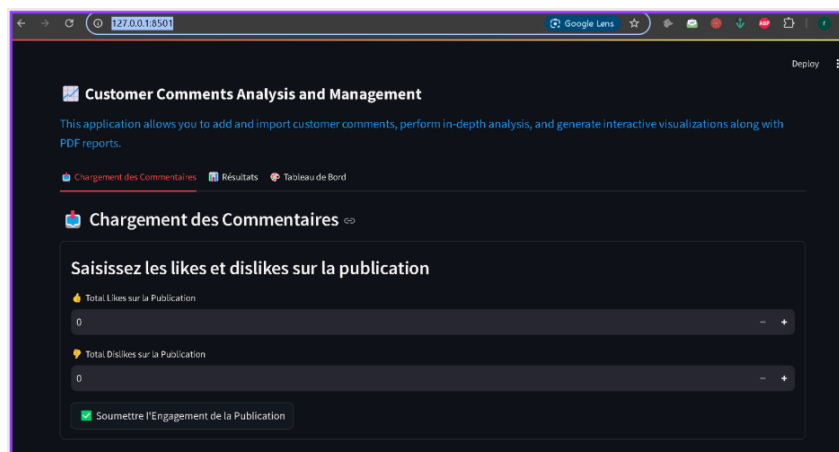


FIGURE 3.9 – l'application sur le localhost

3.4 Déploiement Continu avec GitHub Actions



FIGURE 3.10 – Logo de GitHub

Pour automatiser et simplifier le déploiement de notre application, nous avons intégré un pipeline CI/CD en utilisant **GitHub Actions**. Cet outil permet de configurer des workflows automatisés pour tester, construire, et déployer l'application à chaque mise à jour du code source.

3.4.1 Configuration du Workflow GitHub Actions

Le fichier de configuration `.github/workflows/deploy.yml` contient les étapes nécessaires pour :

1. Vérifier et tester le code source en exécutant les tests.
2. Construire l'image Docker de l'application à l'aide du fichier `Dockerfile`.
3. Déployer l'application sur un environnement de test ou de production.

Voici un exemple simplifié du fichier `deploy.yml` :

```
name: Deploy Streamlit App

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      # Étape 1 : Vérifier le code source
      - name: Checkout code
        uses: actions/checkout@v3

      # Étape 2 : Configuration de Docker
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2

      # Étape 3 : Se connecter à DockerHub
      - name: Log in to DockerHub
        uses: docker/login-action@v2
        with:
          username: ${ secrets.DOCKER_USERNAME }
          password: ${ secrets.DOCKER_PASSWORD }

      # Étape 4 : Construire et pousser l'image Docker
      - name: Build and push Docker image
        uses: docker/build-push-action@v4
        with:
          context: .
          push: true
          tags: ${ secrets.DOCKER_USERNAME }/streamlit-app:latest

      # Étape 5 : Déployer l'application
      - name: Deploy Streamlit App
        run: |
          docker pull ${ secrets.DOCKER_USERNAME }/streamlit-app:latest
          docker run -d -p 8501:8501 ${ secrets.DOCKER_USERNAME }/streamlit-app:latest
```

3.4.2 Avantages de GitHub Actions pour le Projet

En intégrant GitHub Actions dans notre projet, nous avons pu :

- Automatiser les processus répétitifs, comme les tests et le déploiement.
- Réduire le temps de mise en production de nouvelles fonctionnalités.
- Garantir que le déploiement de l'application est toujours effectué dans un environnement fiable et cohérent.

Cette intégration renforce les pratiques DevOps en améliorant la collaboration et la qualité des livrables grâce à des processus automatisés et reproductibles.

Conclusion

Ce chapitre a détaillé la mise en œuvre pratique du projet, en mettant l'accent sur le développement de la plateforme interactive avec Streamlit et la containerisation de l'application avec Docker. Ces outils ont permis de créer une application robuste, portable et facile à déployer, facilitant ainsi l'analyse et l'interprétation des commentaires clients.

Conclusion Générale

Ce projet a démontré l'importance de l'intégration des données textuelles et des interactions sociales pour une analyse approfondie des retours clients. En combinant des approches d'apprentissage automatique, des visualisations intuitives et des outils DevOps, nous avons développé une solution efficace et déployable pour l'évaluation des commentaires.

La capacité de prédire des scores et de fournir des analyses globales sur les publications constitue un atout majeur pour les entreprises souhaitant exploiter la voix du client. Cette démarche s'inscrit dans une perspective d'amélioration continue, où les technologies avancées jouent un rôle clé dans la transformation des données en décisions stratégiques.

Références

1. Scikit-learn : Machine Learning in Python, <https://scikit-learn.org/>
2. TensorFlow : An end-to-end open-source platform for machine learning, <https://www.tensorflow.org/>
3. Natural Language Toolkit (nltk), <https://www.nltk.org/>
4. Streamlit : The fastest way to build and share data apps, <https://streamlit.io/>
5. WordCloud for Python : https://github.com/amueller/word_cloud
6. Plotly : Python graphing library, <https://plotly.com/python/>
7. Docker Documentation : <https://docs.docker.com/>
8. FPDF : Free PDF generation library for Python, <https://pyfpdf.github.io/fpdf2/>
9. Kaggle : Amazon Fine Food Reviews Dataset, <https://www.kaggle.com/snap/amazon-fine-food-reviews>