Projet : Analyse et Classification d'Articles de Presse

Kuassi Pierre DOVODJI

2025-01-20

Contents

1 Introduction

L'essor des médias numériques a entraîné une explosion de la quantité d'articles publiés chaque jour sur des sujets variés comme l'économie, le sport et la science. Face à ce flot d'informations, il est crucial pour les organisations de presse de pouvoir classer efficacement ces articles en fonction de leur thématique principale.

Ce projet vise à développer une solution automatisée pour classifier ces articles dans des catégories telles qu'Économie, Sport et Science. Pour ce faire, nous avons conçu un modèle de machine learning capable d'analyser les données textuelles des articles et de les attribuer à la bonne catégorie.

L'objectif final est de déterminer la méthode la plus performante pour classifier un article, qu'il s'agisse du titre, du titre et de la description, ou de l'article complet. Ce rapport détaille les choix méthodologiques, les résultats obtenus et propose des pistes d'amélioration pour optimiser les performances du modèle.

- **Présentation des données**

La base de données utilisée pour ce projet contient un ensemble d'articles classés en trois catégories principales : **Sport**, **Science**, et **Économie**. Chaque article est structuré en trois champs distincts : le **titre**, la **description**, et le **corps complet** de l'article. La répartition des catégories au sein de l'échantillon est approximativement la suivante :

- **Sport** : 34,72 % des articles
- **Science** : 32,79 % des articles
- **Économie** : 32,49 % des articles

Cela reflète une répartition assez équilibrée entre les trois thématiques, ce qui permet de développer et de tester un modèle de classification supervisée dans un contexte varié.

Les articles sont présentés sous forme de texte brut, et chaque champ (titre, description, et corps de l'article) contient des informations spécifiques qui pourront être utilisées pour alimenter les modèles de classification. Le modèle sera chargé de déterminer la catégorie appropriée (Sport, Science ou Économie) pour chaque article en fonction des éléments textuels fournis.

La base de données contient au total **3681 articles**, qui seront divisés en deux ensembles : un ensemble d'entraînement (70%) et un ensemble de test (30%), afin d'évaluer la performance du modèle.

2 Exploration et Prétraitement des Données

Avant d'entraîner nos modèles de classification, il est essentiel de comprendre la structure et les caractéristiques de notre corpus d'articles. Une fois les fichiers chargés et la donnée créée, nous avons commencé par effectuer un prétraitement rigoureux afin de préparer le texte pour les analyses ultérieures. Cette étape inclut le nettoyage des données, où nous avons supprimé les caractères indésirables, normalisé les espaces et utilisé la tokenisation pour diviser les textes en mots individuels. Ensuite, nous avons procédé à une analyse descriptive des articles, afin de mieux comprendre les caractéristiques des textes, telles que la fréquence des termes ou leur distribution. Nous avons également éliminé les **stop words**, qui sont des mots courants sans valeur informative significative (comme "le", "la", "être"), afin de ne garder que les termes importants pour l'analyse. Enfin, une analyse approfondie du contenu des articles a été réalisée pour extraire les motifs sous-jacents et préparer les données pour la modélisation.

2.1 Analyse de la longueur des articles

Voici le tableau complété avec les informations sur le nombre de mots distincts par catégorie :

Catégorie	Longueur moyenne (caractères)	Longueur moyenne (mots)	Nombre de mots distincts
Économie	183.82	414.82	29 197
Science	186.87	417.58	35 619
Sport	197.07	449.82	37 712

On observe que les articles de la catégorie **Sport** sont légèrement plus longs que ceux des catégories **Économie** et **Science**.

Visualisation :

Nous avons représenté cette distribution avec des **boxplots** afin de mieux visualiser la variation des longueurs des articles.

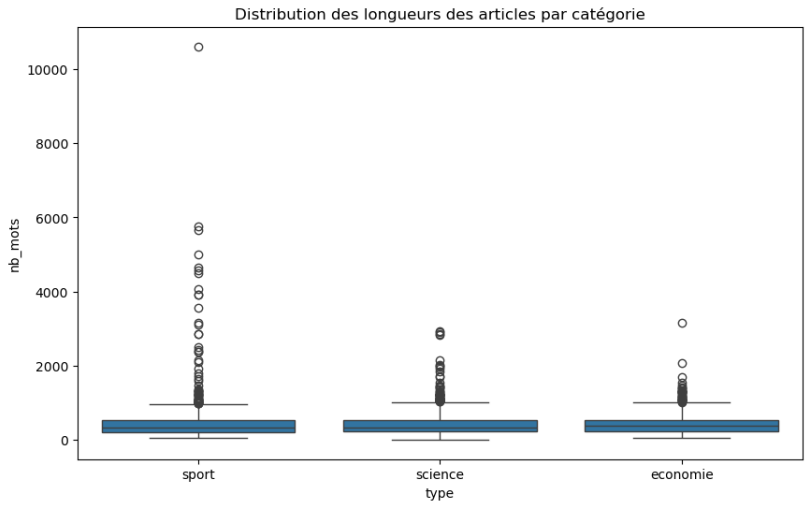

Figure 1: Boîte à moustaches par catégorie

On voit bien qu'il y un mot dans la catégorie sport qui est très fréquent c'est le mot `france`.

2.2 Fréquence des mots par catégorie

L'extraction des mots les plus fréquents dans chaque catégorie permet de mettre en évidence les thématiques dominantes :

Top	sport	science	economie
1	(france, 1435)	(terre, 758)	(france, 1149)
2	(paris, 1144)	(mission, 610)	(gouvernement, 923)
3	(monde, 1018)	(etait, 584)	(deuros, 799)
4	(finale, 925)	(spatiale, 569)	(millions, 737)
5	(olympique, 841)	(premiere, 541)	(ministre, 710)
6	(match, 785)	(france, 508)	(euros, 648)
7	(francais, 715)	(scientifiques, 496)	(decembre, 636)
8	(place, 674)	(premier, 466)	(groupe, 611)
9	(olympiques, 634)	(chercheurs, 452)	(milliards, 588)
10	(premier, 619)	(annees, 437)	(budget, 554)

Visualisation :

Afin de permetre d'identifier visuellement les termes récurrents et leur importance relative nous avons généré des **word clouds** et **histogramme** pour chaque catégorie d'article.

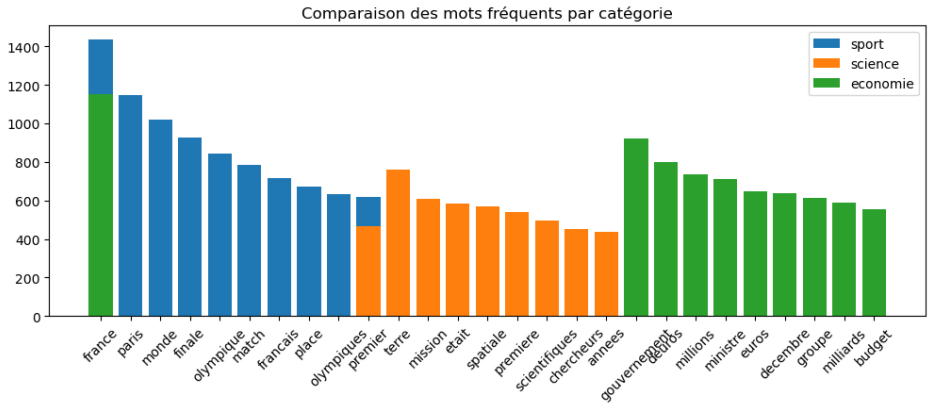

Figure 2: histogramme des top mots

Figure 3: Word clouds par catégorie d'article

2.3 Analyse des mots-clés caractéristiques

En utilisant la pondération **TF-IDF**, nous avons extrait les termes les plus caractéristiques du corpus :

Top	sport	science	economie
0	(france, 0.0141)	(terre, 0.0103)	(france, 0.0141)
1	(paris, 0.0126)	(mission, 0.0098)	(gouvernement, 0.0128)
2	(monde, 0.0109)	(spatiale, 0.0092)	(ministre, 0.0107)
3	(finale, 0.0106)	(france, 0.0081)	(deuros, 0.0107)
4	(match, 0.0100)	(fusee, 0.0077)	(euros, 0.0103)
5	(francais, 0.0097)	(premiere, 0.0077)	(budget, 0.0103)
6	(paralympiques, 0.0090)	(lespace, 0.0073)	(millions, 0.0098)
7	(olympique, 0.0089)	(scientifiques, 0.0070)	(decembre, 0.0095)
8	(olympiques, 0.0081)	(premier, 0.0066)	(milliards, 0.0092)
9	(vendredi, 0.0077)	(spacex, 0.0066)	(groupe, 0.0089)

Visualisation :
Un **histogramme des scores TF-IDF** a été généré pour illustrer les termes les plus distinctifs du corpus.

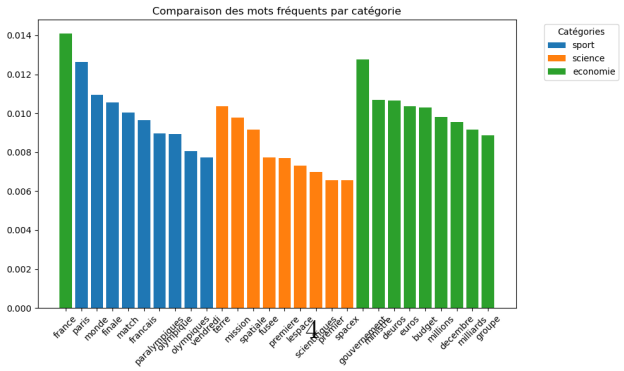

Figure 4: histogramme des scores TF-IDF

2.4 Comparaison globale du vocabulaire

Nous avons analysé la diversité lexicale entre les catégories en comparant les mots uniques et les termes communs entre les domaines **Sport, Science et Économie**.

- **Termes spécifiques :**
  - *Sport* : olympique, finale, match, paralympiques, olympiques, vendredi
  - *Science* : mission, spatiale, chercheurs, scientifiques, lespace
  - *Économie* : marché, croissance, ministres, milliards, budget, euros, décembre.
- **Termes en commun :**
  - france, premier, monde, place, groupe

Ces résultats montrent que bien que certaines thématiques soient transversales, chaque catégorie possède un champ lexical distinct.

3 Méthodologie de Classification

Dans cette section, nous détaillons la méthodologie utilisée pour la classification des articles en fonction de leur catégorie (Économie, Science, Sport). La procédure implique l'utilisation de plusieurs modèles d'apprentissage supervisé, l'optimisation des hyperparamètres à l'aide de **RandomizedSearchCV**, ainsi que l'évaluation de la performance des modèles à travers différentes métriques.

3.1 Définition des scénarios de classification

1. **C1 : Basé sur le titre** Dans ce scénario, la classification des articles est effectuée uniquement en utilisant le **titre** de l'article. Le titre d'un article est souvent un résumé condensé de son contenu et peut fournir des indices précieux pour déterminer la catégorie à laquelle il appartient. Ce scénario de classification repose sur l'idée que le titre peut refléter de manière succincte le sujet de l'article.
2. **C2 : Basé sur le titre + description** Ce scénario prend en compte à la fois le **titre** et la **description** de l'article. La description, qui se trouve souvent sous le titre ou dans le résumé de l'article, permet d'enrichir les informations disponibles pour la classification. Elle peut offrir des détails supplémentaires qui clarifient le sujet principal de l'article et donc améliorer la précision du modèle de classification. Ce scénario permet d'exploiter non seulement le titre mais aussi les premiers éléments descriptifs associés à l'article.
3. **C3 : Basé sur le titre + description + texte complet** Le dernier scénario utilise le **titre**, la **description** et le **texte complet** de l'article pour la classification. Ce scénario est le plus complet car il intègre toutes les informations disponibles, offrant une vue d'ensemble du contenu. Le texte complet permet de mieux comprendre le contexte global de l'article, et ainsi de faciliter la classification avec un plus grand niveau de précision. Ce scénario peut être particulièrement utile lorsque les informations dans le titre ou la description ne suffisent pas pour attribuer correctement l'article à sa catégorie.

3.2 Modèles Supervisés

Les modèles sélectionnés pour cette tâche sont des algorithmes d'apprentissage supervisé. Ils ont été choisis en raison de leur popularité et de leur efficacité dans des scénarios de classification textuelle. Voici une description des modèles utilisés :

1. **Régression Logistique (Logistic Regression)**
   - **Description** : La régression logistique est un modèle de classification qui utilise la probabilité pour prédire la classe d'un échantillon. Elle fonctionne bien pour des problèmes de classification où les relations entre les caractéristiques et la classe sont linéaires.
   - **Paramètres optimisés** :
     - `C` : Paramètre de régularisation qui contrôle l'équilibre entre la complexité du modèle et le risque de surapprentissage. Plus `C` est petit, plus la régularisation est forte.

2. **Arbre de Décision (Decision Tree)**
   - **Description** : L'arbre de décision est un modèle supervisé qui sépare les données en fonction des caractéristiques afin de prédire la classe. Il divise l'espace des caractéristiques en régions homogènes par rapport à la classe cible.
   - **Paramètres optimisés** :
     - `max_depth` : Profondeur maximale de l'arbre. Un arbre trop profond peut surapprendre les données.
     - `min_samples_split` : Nombre minimum d'échantillons requis pour diviser un nœud.
     - `min_samples_leaf` : Nombre minimum d'échantillons dans une feuille de l'arbre.
3. **Forêt Aléatoire (Random Forest)**
   - **Description** : La forêt aléatoire est un ensemble d'arbres de décision où chaque arbre est construit à partir d'un sous-ensemble aléatoire des données. Ce modèle est robuste face au surapprentissage et génère des prédictions plus stables.
   - **Paramètres optimisés** :
     - `n_estimators` : Nombre d'arbres dans la forêt.
     - `max_depth`, `min_samples_split`, `min_samples_leaf` : Paramètres similaires à ceux de l'arbre de décision.
     - `bootstrap` : Indicateur de bootstrap pour échantillonner les données avec remplacement.
4. **Gradient Boosting**
   - **Description** : Le gradient boosting est une méthode d'ensemble qui construit des arbres de décision successivement, chaque arbre corrigeant les erreurs du précédent. Il est particulièrement puissant pour des modèles complexes.
   - **Paramètres optimisés** :
     - `n_estimators` : Nombre d'arbres à construire.
     - `learning_rate` : Taux d'apprentissage pour chaque arbre.
     - `max_depth` : Profondeur des arbres.
     - `min_samples_split` : Nombre minimum d'échantillons pour diviser un nœud.
5. **xGBoost**
   - **Description** : xGBoost est une implémentation optimisée de l'algorithme de gradient boosting. Elle est connue pour sa rapidité et sa performance dans des tâches de classification complexes.
   - **Paramètres optimisés** :
     - `n_estimators` : Nombre d'arbres.
     - `learning_rate` : Taux d'apprentissage pour chaque arbre.
     - `max_depth` : Profondeur des arbres.
     - `min_child_weight` : Poids minimum d'un enfant pour créer une division dans l'arbre.
     - `reg_alpha`, `reg_lambda` : Termes de régularisation L1 et L2 pour contrôler la complexité du modèle.
6. **KNN (K-Nearest Neighbors)**
   - **Description** : KNN est un algorithme basé sur la similarité des voisins. Il classe un échantillon en fonction des classes de ses k plus proches voisins dans l'espace des caractéristiques.
   - **Paramètres optimisés** :
     - `n_neighbors` : Nombre de voisins à considérer pour déterminer la classe.
     - `weights` : Méthode de pondération des voisins ("uniforme" ou "distance").
     - `p` : Exposant de la distance (1 pour Manhattan, 2 pour Euclidienne).
7. **SVM (Support Vector Machine)**
   - **Description** : Le SVM est un algorithme puissant pour la classification binaire. Il recherche un hyperplan qui maximise la séparation entre les différentes classes.
   - **Paramètres optimisés** :
     - `C` : Paramètre de régularisation.
     - `kernel` : Type de noyau à utiliser (linéaire, RBF, polynomial, sigmoïde).
     - `gamma` : Paramètre de régularisation pour certains types de noyaux, influençant la complexité du modèle.
8. **Naïve Bayes Multinomial (MultinomialNB)**
   - **Description** : Le Naïve Bayes multinomial est un modèle probabiliste qui est bien adapté aux

problèmes de classification textuelle, où les caractéristiques sont représentées sous forme de comptages de mots.

- **Paramètres optimisés** :
  - `alpha` : Paramètre de régularisation pour éviter le surapprentissage.
  - `fit_prior` : Si vrai, le modèle tiendra compte de la distribution des classes dans les données.

3.3 Optimisation des Modèles

L'optimisation des hyperparamètres est effectuée à l'aide de **RandomizedSearchCV**, ce qui permet de rechercher efficacement dans un espace d'hyperparamètres large et de trouver les meilleurs paramètres pour chaque modèle. Le paramètre `n_iter=10` spécifie le nombre d'itérations de rechrche aléatoire, et **cv=5** indique une validation croisée à 5 plis. Le choix de RandomizedSearchCV pour l'optimisation des hyperparamètres des modèles a été fait principalement pour réduire le temps d'inférence tout en permetant une exploration efficace des espaces d'hyperparamètres.

Chaque modèle est évalué sur l'ensemble d'entraînement et sur l'ensemble de test, et ses performances sont mesurées à l'aide des **métriques** suivantes : **accuracy**, **F1-score**, **recall**, et **precision**. Les résultats sont enregistrés pour permettre une comparaison et une sélection du meilleur modèle. Et les meilleurs modèles des trois sénarions sont comparés aussi sur la matrice de confusion et la courbe ROC.

3.4 Évaluation des modèles

3.4.1 Accuracy (Précision)

L'accuracy, ou précision, est l'une des métriques les plus simples et les plus utilisées. Elle est calculée comme le rapport entre le nombre de prédictions correctes et le nombre total de prédictions :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Cette métrique donne une idée générale de la performance du modèle. Toutefois, elle peut être biaisée dans le cas où les données sont déséquilibrées (par exemple, si certaines catégories sont beaucoup plus fréquentes que d'autres).

3.4.2 F1-score

Le F1-score est une métrique qui combine la **précision** et le **rappel** en une seule mesure, particulièrement utile lorsque les classes sont déséquilibrées. Il est calculé comme la moyenne harmonique entre la précision et le rappel :

$$\text{F1-score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Le F1-score donne un équilibre entre ces deux métriques et est particulièrement précieux lorsque nous voulons éviter à la fois de fausses positives et de fausses négatives.

3.4.3 Recall (Rappel)

Le **rappel** mesure la capacité du modèle à identifier toutes les instances positives dans les données. Il est défini comme le ratio des prédictions positives correctes par rapport au nombre total d'instances positives réelles :

$$\text{Recall} = \frac{\text{Vrai Positifs}}{\text{Vrai Positifs} + \text{Faux Négatifs}}$$

Le rappel est crucial dans des situations où il est plus important de détecter toutes les instances positives, même au prix d'un certain nombre de fausses positives.

3.4.4 Précision

La **précision** mesure la capacité du modèle à prédire uniquement des instances positives correctes. Elle est calculée comme le ratio des prédictions positives correctes par rapport au nombre total de prédictions positives :

$$\text{Precision} = \frac{\text{Vrai Positifs}}{\text{Vrai Positifs} + \text{Faux Positifs}}$$

Cette métrique est importante dans des scénarios où il est plus important de réduire le nombre de fausses alertes, même si cela signifie que certaines instances positives seront manquées.

3.4.5 Matrice de confusion

La **matrice de confusion** est un outil qui permet de visualiser les performances d'un modèle en termes de prédictions correctes et incorrectes, classées selon les différentes catégories. Elle contient quatre valeurs clés :

- **Vrai Positifs (VP)** : Le modèle a correctement prédit une instance positive.
- **Faux Positifs (FP)** : Le modèle a incorrectement prédit une instance positive (fausse alerte).
- **Vrai Négatifs (VN)** : Le modèle a correctement prédit une instance négative.
- **Faux Négatifs (FN)** : Le modèle a incorrectement prédit une instance négative.

La matrice de confusion permet d'examiner la répartition des erreurs du modèle, ce qui est essentiel pour comprendre où il fait des prédictions incorrectes et pour ajuster le modèle.

3.4.6 Courbe ROC (Receiver Operating Characteristic)

La courbe **ROC** est un graphique qui permet de visualiser la performance d'un modèle en fonction de différents seuils de classification. Elle trace le **taux de vrais positifs (TPR)** contre le **taux de faux positifs (FPR)** pour plusieurs seuils de probabilité.

- **Taux de Vrais Positifs (TPR)** : Il s'agit du rappel, qui est la proportion de vrais positifs parmi les instances positives réelles.

$$\text{TPR} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

- **Taux de Faux Positifs (FPR)** : Il s'agit du ratio des faux positifs parmi les instances négatives réelles.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{VN}}$$

La **courbe ROC** est particulièrement utile pour comparer plusieurs modèles, car un modèle qui se situe plus près du coin supérieur gauche de la courbe est plus performant. Un modèle parfait aurait une courbe qui passe par ce coin, avec un TPR de 1 et un FPR de 0.

3.5 Transformation des données

Pour réaliser la classification, il est essentiel de transformer les données textuelles en une représentation numérique que les modèles d'apprentissage automatique peuvent comprendre. Plusieurs méthodes de vectorisation sont disponibles, mais nous avons choisi d'utiliser le **TfidfVectorizer**. Cette technique produit

une matrice creuse où chaque ligne représente un document (un article ici), et chaque colonne correspond à un mot unique du vocabulaire.

Le **TF-IDF** (Term Frequency - Inverse Document Frequency) repose sur deux éléments principaux : 1. **La fréquence du terme (TF)**, qui mesure le nombre d'occurrences d'un mot dans un document donné. Un mot plus fréquemment présent dans un document aura une valeur TF plus élevée. 1. **L'inverse de la fréquence des documents (IDF)**, qui évalue l'importance d'un mot à travers l'ensemble du corpus. Si un mot apparaît dans de nombreux documents, son IDF sera faible, car il est considéré comme peu informatif. En revanche, un mot qui n'apparaît que dans quelques documents aura un IDF élevé, mettant en évidence son importance contextuelle.

Le **TfidfVectorizer** convertit chaque document en un vecteur numérique où chaque valeur indique l'importance d'un mot dans un article par rapport au corpus total. En choisissant un **ngram_range=(1, 2)**, nous générons à la fois des unigrammes (mots individuels) et des bigrammes (paires de mots consécutifs), permettant ainsi de capturer des relations entre les mots.

4 Analyse des Résultats

4.1 Scénario C1 : Le titre de l'article

Après la vectorisation des textes, nous avons obtenu une matrice de caractéristiques de dimension (3681, 23981), représentant le nombre total de termes uniques sélectionnés par TF-IDF.
L'optimisation des modèles a été réalisée. L'entraînement a duré environ 7 minutes. Les performances des différents modèles sont consignées dans lex tableaux suivants :

- **Train**

Modèle	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.97205	0.97209	0.972531	0.97205
Decision Tree	0.612966	0.581825	0.780551	0.612966
Random Forest	0.854814	0.856247	0.88217	0.854814
Gradient Boosting	0.971661	0.971659	0.971891	0.971661
xGBoost	0.783385	0.7853	0.837958	0.783385
KNN	0.97205	0.97207	0.972353	0.97205
SVM	0.97205	0.97208	0.97243	0.97205
MultinomialNB	0.971273	0.971274	0.971773	0.971273

- **Test**

Modèle	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.861538	0.861603	0.863033	0.861538
Decision Tree	0.571946	0.537275	0.745674	0.571946
Random Forest	0.771946	0.773689	0.812922	0.771946
Gradient Boosting	0.81991	0.820124	0.828645	0.81991
xGBoost	0.725792	0.726475	0.802911	0.725792
KNN	0.485068	0.42117	0.704799	0.485068
SVM	0.860633	0.860674	0.861638	0.860633
MultinomialNB	0.861538	0.861284	0.861621	0.861538

- **Interprétation :**

Le meilleur modèle est la **Régression Logistique**. Il a le **meilleur score** sur les données de test (86,2%), ce qui veut dire qu'il **généralise bien**. Et aussi nécessite moins de temps d'entrainement comparativement a certiant modèle.

Le **Gradient Boosting** et **Multinomial Naïve Bayes** sont aussi de bons choix.

Par contre, **l'Arbre de Décision et KNN** fonctionnent **moins bien**, ils ne généralisent pas bien.

- **Matrice de confusion, courbe ROC de la regression logistique et Probabité de prédictition**

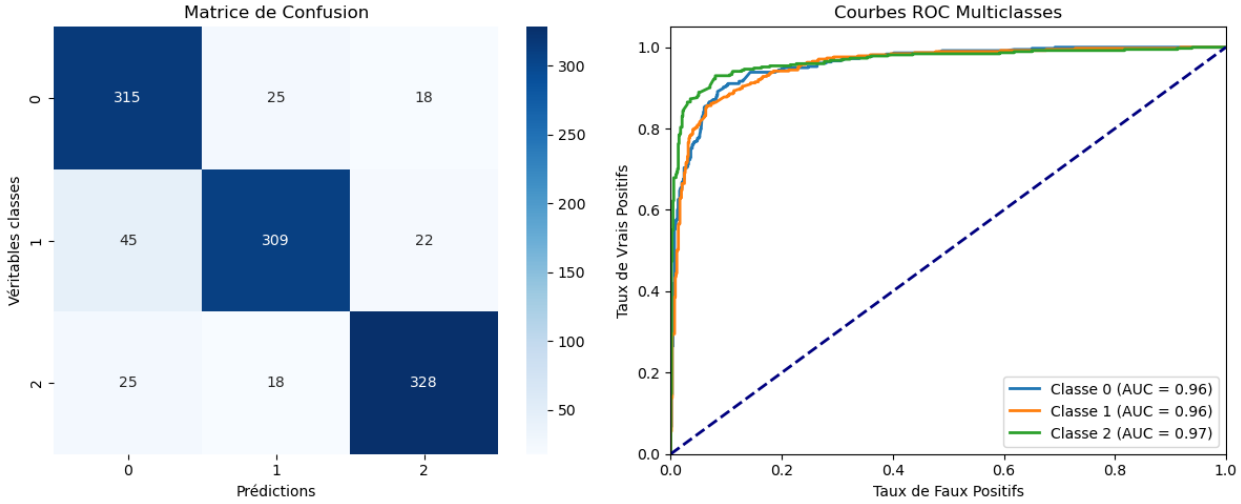

Figure 5: Matrice de confusion et courbe ROC sénario 1

Le modèle fait de bonnes prédictions avec peu d'erreurs. La matrice de confusion montre qu'il reconnaît bien les différentes catégories. Les courbes ROC montrent qu'il distingue bien les classes, avec des scores élevés (autour de 0.96 et 0.97). Cela signifie que le modèle est fiable et performant pour classer les données.

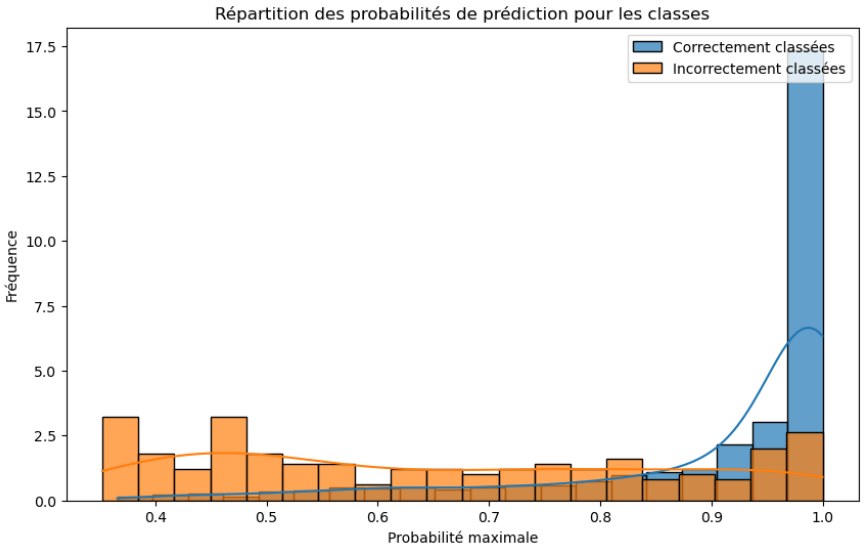

Figure 6: Probabité de prédictition sénario 1

4.2 Scénario C2: Le titre de l'article et la description de l'article

Lors de la vectorisation des données textuelles, la première approche a donné une matrice de taille très grande, avec 71310 colonnes. Cela a entraîné un temps d'entraînement long et des problèmes de mémoire. Pour améliorer cela, nous avons réduit la taille de la matrice en éliminant les mots trop fréquents, ceux trop rares, et en limitant le nombre de caractéristiques utilisées. Nous avons ainsi fixé un seuil de fréquence maximale (`max_df`) à 95 %, ce qui a permis d'éliminer les mots présents dans plus de 95 % des documents, jugés trop communs pour apporter de l'information utile. Nous avons aussi fixé un seuil de fréquence minimale (`min_df`) à 0,2 %, excluant les mots présents dans moins de 0,2 % des documents, considérés comme trop rares. Après cette optimisation, la matrice a 2878 colonnes. Cette réduction a permis de diminuer le temps d'entraînement tout en maintenant des performances similaires à celles de la première approche, avec une différence d'erreur de moins de 0,1%. Cela montre qu'il est possible d'améliorer l'efficacité du modèle sans perdre en qualité. Voici les résultats obtenus :

- **Train**

Modèle	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.968556	0.968584	0.968877	0.968556
Decision Tree	0.710792	0.704629	0.804351	0.710792
Random Forest	0.911102	0.912409	0.925555	0.911102
Gradient Boosting	0.97205	0.972087	0.972479	0.97205
xGBoost	0.903727	0.904349	0.90963	0.903727
KNN	0.971273	0.971274	0.972005	0.971273
SVM	0.97205	0.972083	0.97243	0.97205
MultinomialNB	0.938665	0.938773	0.939633	0.938665

- **Test**

Modèle	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.880543	0.880617	0.880751	0.880543
Decision Tree	0.657014	0.649469	0.744578	0.657014
Random Forest	0.840724	0.84199	0.852143	0.840724
Gradient Boosting	0.877828	0.877953	0.878219	0.877828
xGBoost	0.846154	0.847388	0.854563	0.846154
KNN	0.515837	0.484964	0.725569	0.515837
SVM	0.880543	0.880388	0.880448	0.880543
MultinomialNB	0.882353	0.882553	0.88373	0.882353

- **Interprétation :**
- Logistic Regression, Gradient Boosting, SVM, et KNN sont très bons avec des scores similaires entre l'entraînement et le test, surtout pour la régression logistique et SVM.
- Decision Tree a une grande différence entre l'entraînement et le test, ce qui indique un possible surapprentissage.
- Random Forest et xGBoost sont également bons, mais leur performance diminue légèrement sur le test. Ce qui peux être du a un surapprentissage.

On garde la regression logistique comme meilleur modèle en tenant compte du temps d'inference.

- **Matrice de confusion, courbe ROC de la regression logistique et Probabité de prédictition**

Le modèle fait de bonnes prédictions avec peu d'erreurs. La matrice de confusion montre qu'il reconnaît bien les différentes catégories. Les courbes ROC montrent qu'il distingue bien les classes, avec des scores élevés (autour de 0.97 et 0.98). Cela signifie que le modèle est fiable et performant pour classer les données.

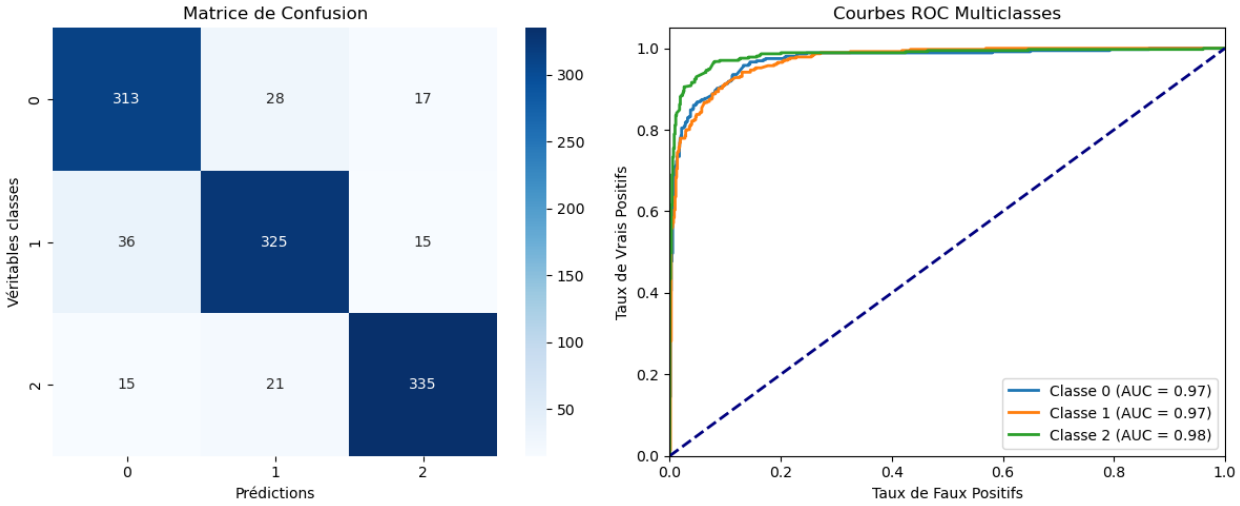

Figure 7: Matrice de confusion et courbe ROC sénario 2

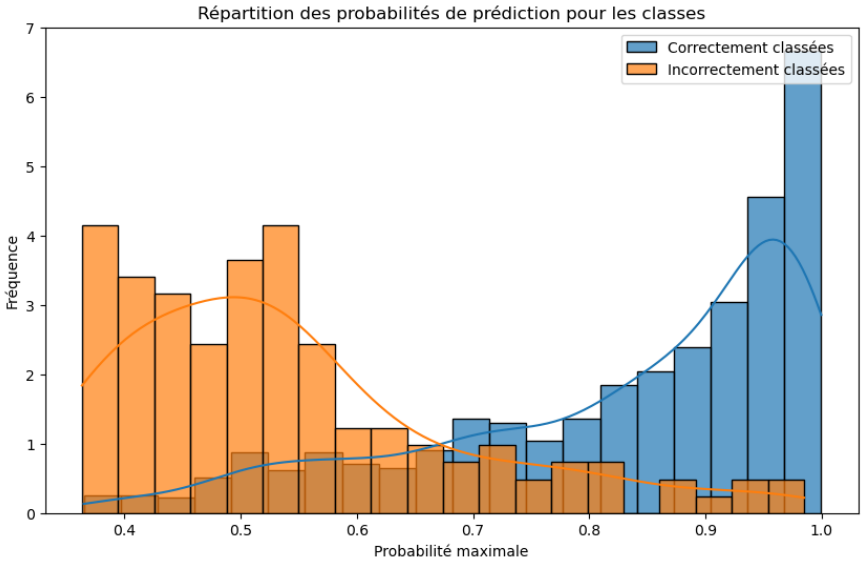

Figure 8: Probabité de prédictition sénario 2

4.3 Scénario C3: Le titre de l'article, la description de l'article, l'article en entier

Dans cette approche, nous avons appliqué la même méthode que dans le scénario 2 avec les même paramètres. Après cette optimisation, la taille de la matrice a 17531 colonnes. Malgré ca l'optimisation et l'entrainement ont durée environ 25 minutes ce qui était de 2heures avant, sans oublier le fait qu'on a pas optimisé tous les paramètres optimiser pour les autres sénario. Mais nous avons obtenu quasiment les même valeurs pour les métriques dans les deux cas. Cette réduction a permis de diminuer le temps d'entraînement tout en conservant des performances comparables à celles du scénario précédent.

- **Train**

Modèle	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.97205	0.972091	0.972582	0.97205
Decision Tree	0.889752	0.890586	0.898331	0.889752
Random Forest	0.947593	0.947738	0.949092	0.947593
Gradient Boosting	0.97205	0.972084	0.972477	0.97205
xGBoost	0.97205	0.972084	0.972528	0.97205
KNN	0.852484	0.845592	0.886303	0.852484
SVM	0.97205	0.972091	0.972582	0.97205
MultinomialNB	0.951475	0.951529	0.952506	0.951475

- **Test**

Modèle	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.920362	0.920313	0.920702	0.920362
Decision Tree	0.78733	0.788404	0.798523	0.78733
Random Forest	0.895928	0.896243	0.897251	0.895928
Gradient Boosting	0.918552	0.918477	0.918586	0.918552
xGBoost	0.901357	0.901248	0.901273	0.901357
KNN	0.61991	0.608397	0.753067	0.61991
SVM	0.923077	0.923028	0.923603	0.923077
MultinomialNB	0.912217	0.912458	0.913488	0.912217

- **Interprétation**

Les modèles SVM et regression logistique ont les meilleurs. Sur l'entraînement, il a des scores très élevés et reste solide sur le test. Ils surpassent les autres modèles comme Random Forest et les autre méthodes. En tenant compte du cout de calcule on garde la regression logistique.

- **Matrice de confusion, courbe ROC de la regression logistique et Probabité de prédictition**

Le modèle fait de bonnes prédictions avec peu d'erreurs. La matrice de confusion montre qu'il reconnaît bien les différentes catégories. Les courbes ROC montrent qu'il distingue bien les classes, avec des scores élevés (autour de 0.98 et 0.99). Cela signifie que le modèle est fiable et performant pour classer les données. Ce

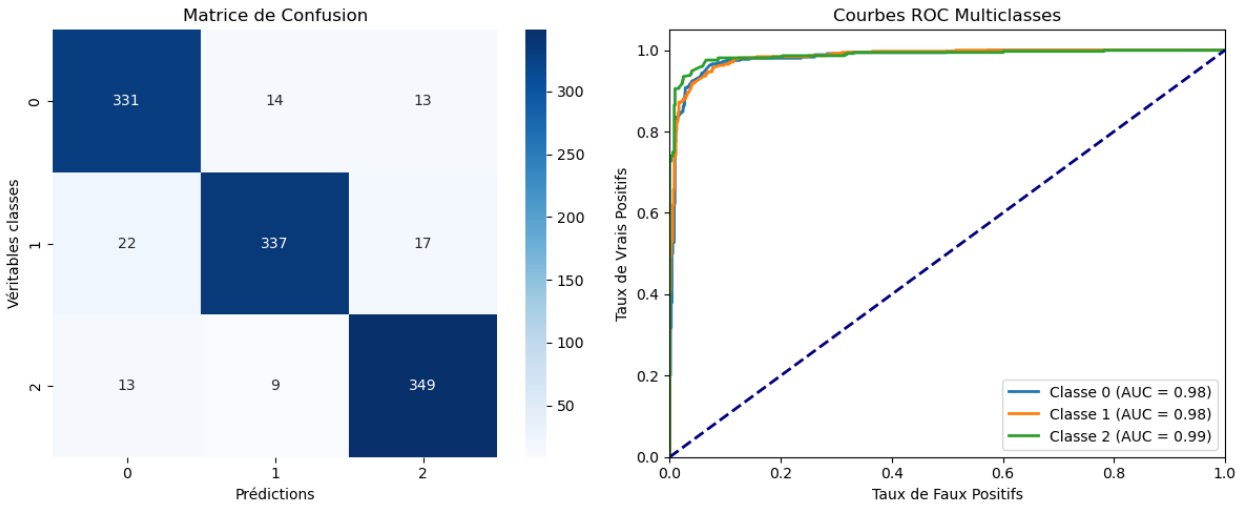

Figure 9: Matrice de confusion et courbe ROC sénario 3

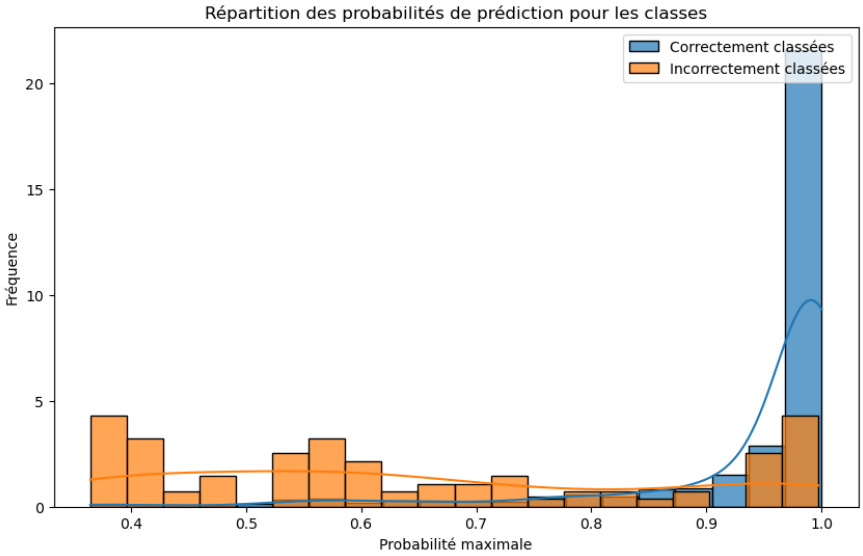

Figure 10: Probabité de prédictition sénario 3

5 Discussion et Perspectives

5.1 Synthèse des résultats

Le scénario C3, bien qu'il améliore légèrement la précision, n'apporte pas de gains significatifs par rapport au scénario C2. La régression logistique dans le scénario C2 montre déjà de bons résultats avec un temps d'entraînement raisonnable. Donc, **le scénario C2** est suffisant et plus efficace, et il n'est pas nécessaire d'utiliser le scénario C3.

5.2 Pistes d'amélioration

- **Augmentation de données** : Une des principales limites de ce projet réside dans la taille du corpus d'articles utilisé pour l'entraînement. L'augmentation de données, par exemple en générant des synonymes ou en utilisant des techniques de paraphrasing pour diversifier les exemples, pourrait permettre de mieux généraliser et de réduire les risques de sur-apprentissage (overfitting).
- **Utilisation d'autres modèles plus avancés** : Bien que les modèles classiques aient montré des résultats intéressants, l'utilisation de modèles plus sophistiqués, comme les réseaux de neurones ou les modèles basés sur les **Transformers** (**BERT**) , pourrait permettre une meilleure compréhension sémantique des textes. Ces modèles ont la capacité de saisir des relations complexes dans les textes et d'améliorer la performance sur des tâches de classification textuelle.
- **Affinement du prétraitement des textes** : Bien que le prétraitement actuel ait permis d'obtenir des résultats décents, un affinage plus poussé pourrait améliorer la performance. Par exemple, l'extraction de n-grams (au-delà des bigrammes), l'utilisation de techniques de lemmatisation au lieu de la simple suppression des stop words ou encore l'intégration de modèles de **word embeddings** (comme **Word2Vec** ou **GloVe**) pourraient enrichir la représentation des textes et conduire à des prédictions plus robustes.
- **Utilisation de techniques de réduction de dimensions** : Des méthodes comme la réduction de dimensionnalité (PCA, LDA) pourraient être explorées pour réduire la taille de la matrice tout en conservant l'information essentielle.

6 Conclusion

Ce projet a permis de développer un système de classification des articles en trois catégories principales : Économie, Science et Sport. En appliquant un prétraitement soigné et en optimisant la représentation des textes, nous avons obtenu des modèles capables de fournir des prédictions fiables et efficaces. Ce travail démontre l'importance d'une bonne préparation des données pour améliorer la précision et la rapidité des classifications, tout en ouvrant la voie à de futures améliorations.