

UNIVERSITÉ DE MONTPELLIER - FACULTÉ DES SCIENCES  
ANNÉE UNIVERSITAIRE 2023 - 2024  
M1 GÉNIE LOGICIEL

## FAKE NEWS DETECTION

Jeudi 2 mai 2024

### Étudiants :

ANESS RABIA  
BILAL BOUSSIHMED  
MOHAMMED DAFAOUI  
ADAM DAIA  
JALAL AZOUZOUT



## 1 Prétraitements et ingénierie de données

En ce qui concerne cette étape, nous avons fusionné les deux fichiers mis à notre disposition, *New\_train* et *New\_test*, afin d'obtenir un ensemble de données plus conséquent.

Une fois fusionnées, nous avons d'abord vérifié s'il y avait des valeurs manquantes dans le dataset. Comme nous avons constaté qu'il manquait quelques valeurs (23), nous avons décidé de supprimer leurs lignes, étant donné que nous disposions d'assez de données pour effectuer les classifications de manière fiable.

Ensuite, nous avons défini une fonction pour automatiser les prétraitements afin de pouvoir les utiliser dans un pipeline de traitement de texte. Cette fonction, `MyCleanText`, regroupe une série de prétraitements avec divers paramètres que nous pouvons activer ou désactiver en fonction des besoins. Les prétraitements disponibles sont les suivants :

- **Conversion en minuscules (lowercase) :**
  - Convertit tous les caractères du texte en minuscules pour uniformiser les données.
- **Suppression des mots vides (removestopwords) :**
  - Élimine les mots courants (tels que "and", "or", "but") qui n'apportent pas de valeur ajoutée significative à l'analyse.
- **Suppression des chiffres (removedigit) :**
  - Supprime tous les chiffres du texte pour se concentrer uniquement sur les mots.
- **Lemmatisation (getlemmatisation) :**
  - Réduit chaque mot à sa forme de base ou canonique. Par exemple, "marchant" et "marché" sont ramenés à "marcher".
- **Nettoyage de la ponctuation :**
  - Supprime tous les caractères de ponctuation pour ne conserver que les mots.
- **Suppression des caractères isolés :**
  - Supprime les caractères qui apparaissent seuls, comme les lettres isolées, pour éviter le bruit dans les données.
- **Tokenisation :**
  - Divise le texte en mots ou tokens pour un traitement plus détaillé.

Plusieurs tests ont été réalisés pour identifier les paramètres les plus pertinents, c'est-à-dire ceux qui produisent les meilleurs résultats.

En matière de vectorisation, nous avons testé la méthode *BagOfWords*, mais nous avons principalement utilisé la méthode *TF-IDF*. Cette dernière permet de réduire l'impact des termes fréquemment présents dans un corpus, lesquels sont moins informatifs pour l'analyse.

Pour obtenir un ensemble de données équilibré, nous avons choisi la méthode de Downsampling. En effet, nous souhaitions éviter de dupliquer les données, ce qui aurait pu entraîner un surapprentissage, ou de générer de nouvelles données potentiellement incorrectes susceptibles de fausser le processus d'apprentissage. Ce choix nous a semblé le plus judicieux pour garantir la fiabilité des résultats.

Nous avons également utilisé la méthode SMOTE pour le suréchantillonnage des classes minoritaires, afin de tester différentes approches pour équilibrer notre jeu de données et améliorer la performance de nos modèles.

## 2 1ère Classification : TRUE vs FALSE

Pour classifier nos données, nous avons d'abord examiné les différentes options disponibles : utiliser le titre, le texte ou une combinaison des deux. Nous avons également exploré divers prétraitements pour déterminer ceux qui produiraient les meilleurs résultats. Enfin, nous avons testé plusieurs classifieurs pour identifier ceux qui conviennent le mieux à notre problématique.

Pour répondre à toutes ces questions, nous avons créé un pipeline intégrant plusieurs classifieurs, notamment : SVC, Logistic Regression, KNN, RandomForest, XGBoost, AdaBoost et GradientBoosting. Nous avons réalisé toutes les combinaisons possibles de prétraitements pour identifier les meilleures options et utilisé GridSearchCV et RandomizedSearchCV pour optimiser les hyperparamètres de chaque classifieur. Une validation croisée avec un nombre de plis (*k-fold*) de 10 a été appliquée pour garantir la robustesse des résultats.

Après avoir effectué tous les tests, nous avons généré un fichier CSV contenant plus de 125 lignes de résultats. Ces résultats ont été analysés pour décider quels classifieurs utiliser, quels prétraitements appliquer et quels hyperparamètres étaient les plus appropriés.

Après analyse, nous avons constaté que les meilleurs résultats étaient obtenus avec les deux prétraitements suivants : `lowercase` et `removestopwords`. La combinaison du texte et du titre a produit les meilleurs résultats, bien que le texte seul ait également donné de bons résultats.

Preprocessing	Classifier	Accuracy	Best Params
lowercase_removestopwords	SVC	0.7728489903424056	{'classifier__C': 1, 'classifier__gamma': 'scale', 'classifier__kernel': 'linear'}
lowercase_removestopwords	RandomForest	0.7489683933274802	{'classifier__max_features': 'sqrt', 'classifier__n_estimators': 200}
lowercase_removestopwords	GradientBoosting	0.73715978928885	{'classifier__learning_rate': 0.2, 'classifier__n_estimators': 100}
lowercase_removestopwords	LogisticRegression	0.7624670763827919	{'classifier__C': 100, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
lowercase_removestopwords	KNN	0.7075065847234416	{'classifier__metric': 'euclidean', 'classifier__n_neighbors': 9, 'classifier__weights': 'distance'}
lowercase_removestopwords	AdaBoost	0.7133450395083406	{'classifier__learning_rate': 0.1, 'classifier__n_estimators': 200}
lowercase_removestopwords	XGBoost	0.7533362598770851	{'classifier__learning_rate': 0.1, 'classifier__max_depth': 5, 'classifier__n_estimators': 300}

FIGURE 1 – Tableau des résultats de la classification TRUE vs FALSE

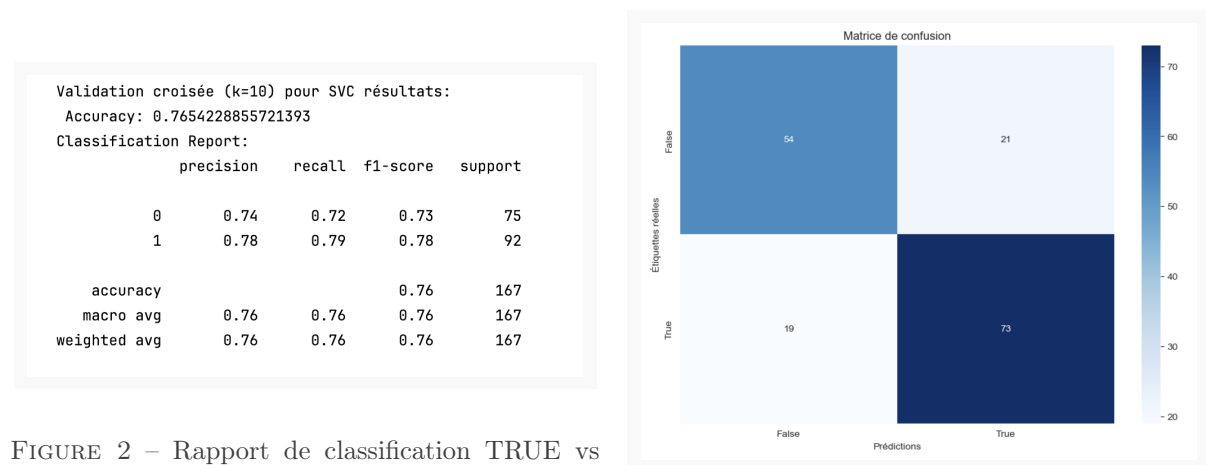


FIGURE 2 – Rapport de classification TRUE vs FALSE

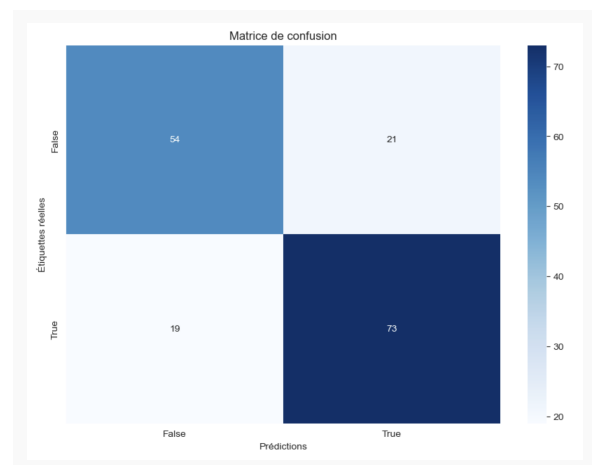


FIGURE 3 – Matrice de confusion TRUE vs FALSE

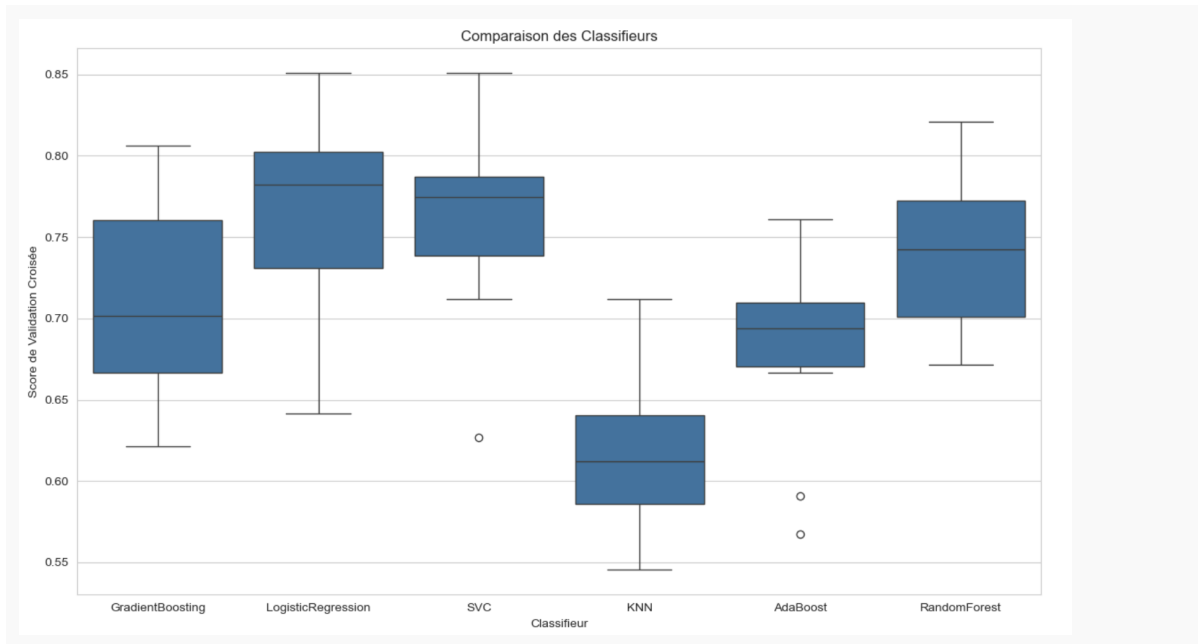


FIGURE 4 – Boxplot des résultats de la classification TRUE vs FALSE

### 3 2ème classification : TRUE and FALSE vs OTHER

Pour la 2ème classification, nous avons regroupé les labels *true* et *false* sous la catégorie 0 et *other* sous la catégorie 1, afin de créer une classification binaire. Cette approche simplifie le modèle et améliore potentiellement les performances de la classification.

Ensuite, nous avons réutilisé notre pipeline pour identifier les meilleurs classifieurs pour notre modèle ainsi que leurs hyperparamètres optimaux. Nous avons continué à utiliser le texte et le titre combinés, ainsi que les prétraitements `lowercase` et `removestopwords`.

Pour tester nos modèles, nous avons réservé 20% de nos données pour la validation. Les résultats obtenus sont les suivants :

Preprocessing	Classifieur	Accuracy	Best Params
lowercase_removestopwords	SVC	0.6572463768115941	{'classifier__C': 10, 'classifier__gamma': 'scale', 'classifier__kernel': 'rbf'}
lowercase_removestopwords	RandomForest	0.640036231884058	{'classifier__max_features': 'sqrt', 'classifier__n_estimators': 100}
lowercase_removestopwords	GradientBoosting	0.6182971014492753	{'classifier__learning_rate': 0.2, 'classifier__n_estimators': 100}
lowercase_removestopwords	LogisticRegression	0.6653985507246376	{'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
lowercase_removestopwords	KNN	0.6143115942028985	{'classifier__metric': 'euclidean', 'classifier__n_neighbors': 5, 'classifier__weights': 'distance'}
lowercase_removestopwords	AdaBoost	0.6048913043478261	{'classifier__learning_rate': 0.1, 'classifier__n_estimators': 200}
lowercase_removestopwords	XGBoost	0.6014492753623188	{'classifier__learning_rate': 0.1, 'classifier__max_depth': 5, 'classifier__n_estimators': 200}

FIGURE 5 – Résultats de la classification TRUE and FALSE vs OTHER

Validation croisée (k=10) pour LogisticRegression - Accuracy: 0.6175724637681159

Classification Report pour LogisticRegression:

	precision	recall	f1-score	support
0	0.60	0.70	0.65	30
1	0.62	0.52	0.57	29
accuracy			0.61	59
macro avg	0.61	0.61	0.61	59
weighted avg	0.61	0.61	0.61	59

FIGURE 6 – Résultats avec Logistic Regression

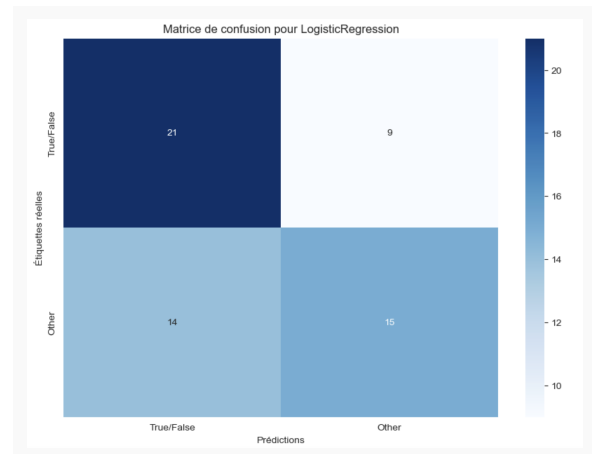


FIGURE 7 – Matrice de confusion TRUE and FALSE vs OTHER

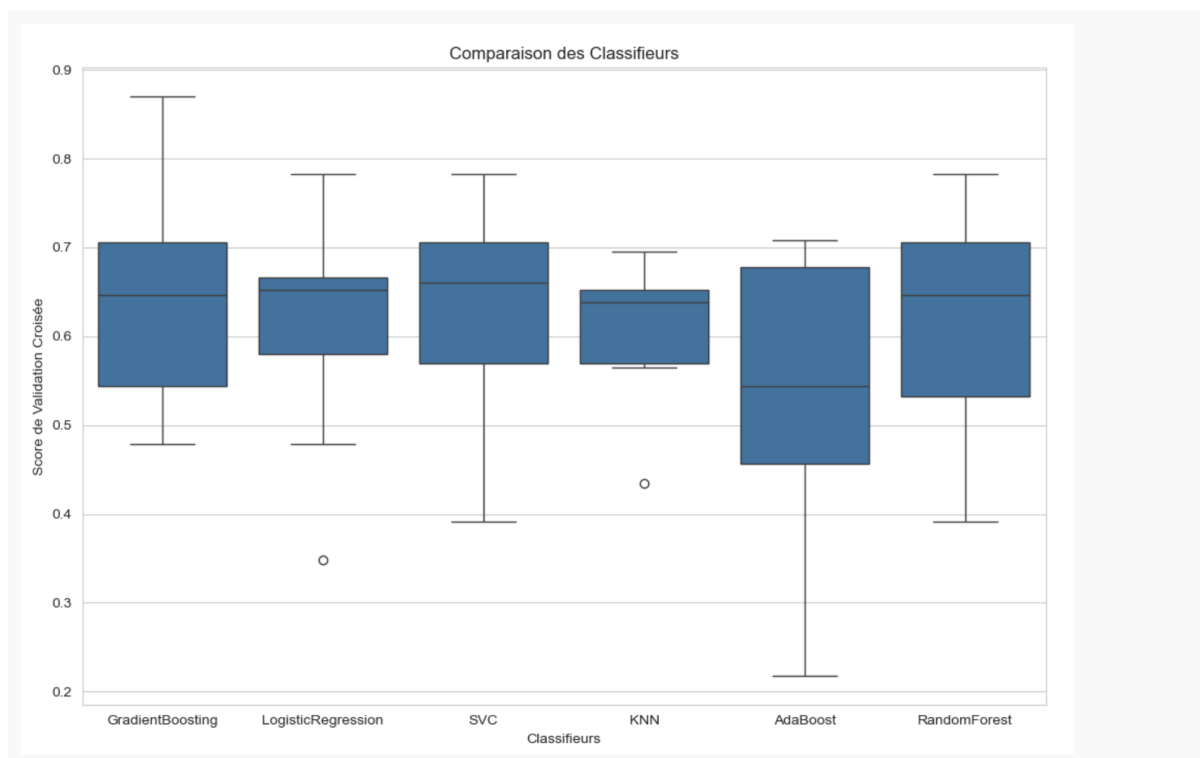


FIGURE 8 – Boxplot des résultats de la classification TRUE and FALSE vs OTHER

Nous constatons que le classifieur *Logistic Regression* s'est révélé être un bon choix pour cette classification, présentant des performances constantes avec une variance réduite. Les autres classifieurs ont également montré des performances intéressantes, mais avec une variance plus élevée, indiquant une stabilité moindre par rapport à *Logistic Regression*.

## 4 3ème classification : TRUE vs FALSE vs MIXTURE vs OTHER

Pour la troisième classification, nous avons suivi les mêmes étapes que pour les deux premières classifications. Nous aurions pu ajouter deux autres prétraitements : *removedigit* et *getlemmatisation*,

car nous avons constaté qu'ils donnent de bons résultats. Cependant, nous avons préféré ne garder que *lowercase* et *removestopwords* car ils offraient une performance légèrement meilleure.

Voici un résumé des résultats de cette classification :

Preprocessing	Classifieur	Accuracy	Best Params
lowercase_removestopwords	SVC	0.49663841807909614	{'classifier__C': 1, 'classifier__gamma': 'scale', 'classifier__kernel': 'rbf'}
lowercase_removestopwords	RandomForest	0.4745197740112994	{'classifier__max_features': 'sqrt', 'classifier__n_estimators': 200}
lowercase_removestopwords	GradientBoosting	0.435819209039548	{'classifier__learning_rate': 0.2, 'classifier__n_estimators': 200}
lowercase_removestopwords	LogisticRegression	0.4831638418079095	{'classifier__C': 1, 'classifier__penalty': 'l2', 'classifier__solver': 'liblinear'}
lowercase_removestopwords	KNN	0.4475988700564971	{'classifier__metric': 'euclidean', 'classifier__n_neighbors': 7, 'classifier__weights': 'distance'}
lowercase_removestopwords	AdaBoost	0.3531920903954803	{'classifier__learning_rate': 0.1, 'classifier__n_estimators': 200}
lowercase_removestopwords	XGBoost	0.46276836158192103	{'classifier__learning_rate': 0.1, 'classifier__max_depth': 7, 'classifier__n_estimators': 300}

FIGURE 9 – Tableau des résultats de la classification TRUE vs FALSE vs MIXTURE vs OTHER

Validation croisée (k=10) pour LogisticRegression - Accuracy: 0.4992464539887892				
Classification Report pour LogisticRegression:				
	precision	recall	f1-score	support
0	0.31	0.33	0.32	27
1	0.46	0.43	0.45	38
2	0.44	0.36	0.40	33
3	0.40	0.48	0.44	29
accuracy			0.40	119
macro avg	0.40	0.40	0.40	119
weighted avg	0.41	0.40	0.40	119

FIGURE 10 – Résultat avec Logistic Regression

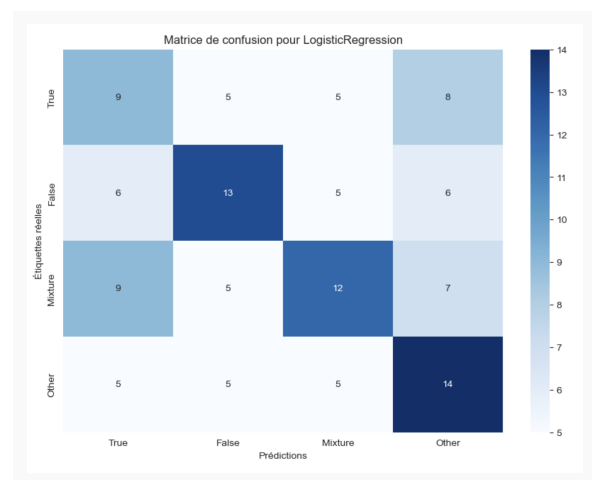


FIGURE 11 – Matrice de confusion TRUE vs FALSE vs MIXTURE vs OTHER

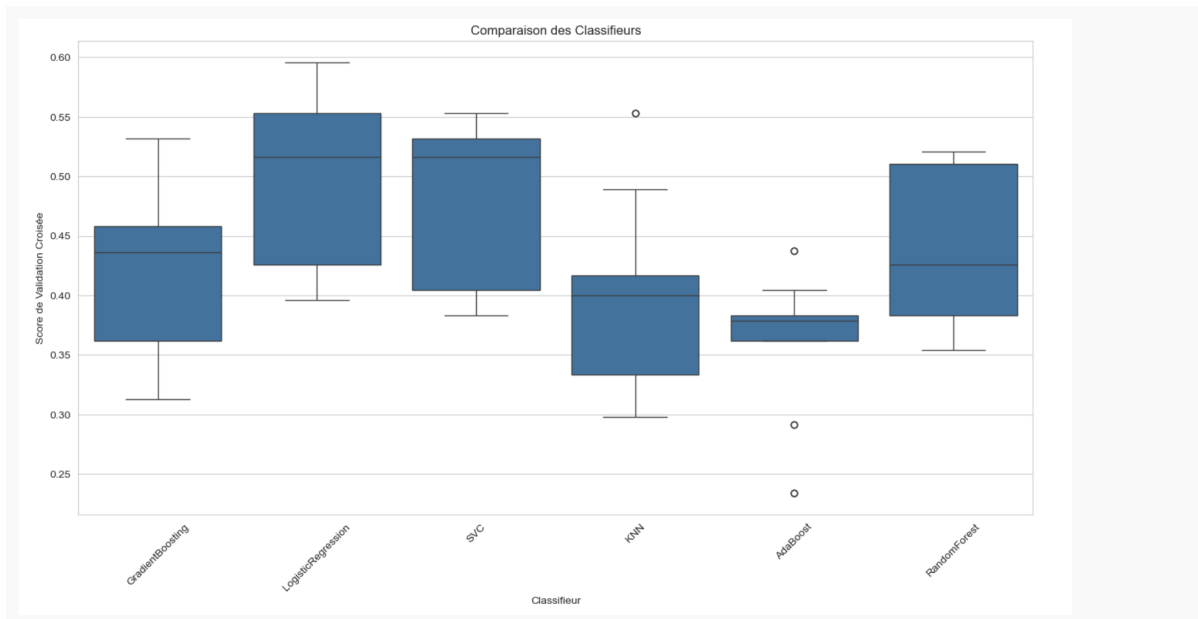


FIGURE 12 – Boxplot TRUE vs FALSE vs MIXTURE vs OTHER

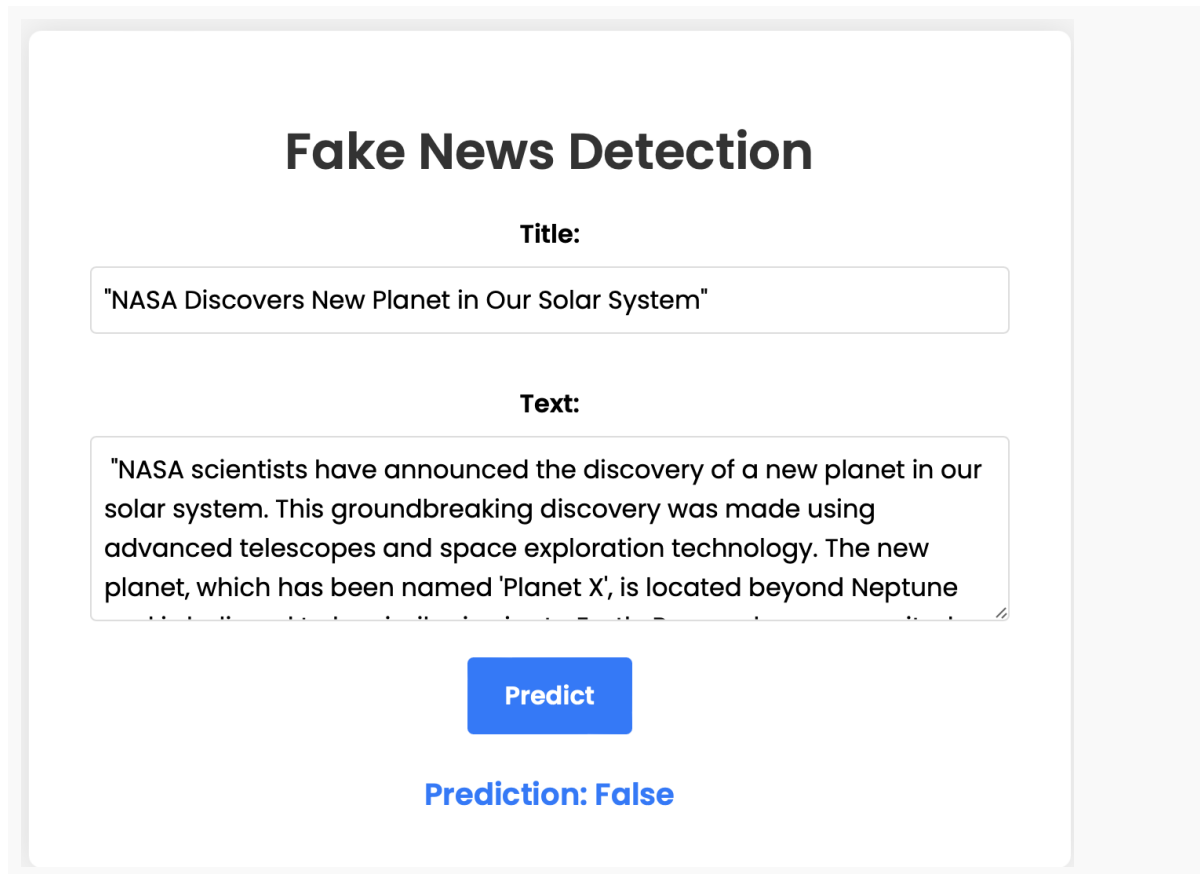
En examinant les résultats, nous constatons que les précisions obtenues ne sont pas optimales, avec des scores inférieurs à 50%. Cela est principalement dû au fait que nous avons utilisé le *downsampling* pour équilibrer les classes. Nous avons également essayé le *upsampling*, ce qui a conduit à des résultats très élevés, atteignant jusqu'à 97%. Cependant, cela a probablement entraîné un phénomène de surapprentissage.

## 5 Déploiement

Après avoir identifié notre modèle le plus performant, à savoir le classifieur SVC pour la classification True vs False, nous avons procédé à son déploiement en utilisant les bibliothèques *Pickle* et *Flask*. Ce déploiement permet de rendre le modèle accessible via une interface web où les utilisateurs peuvent entrer un titre et un texte, et recevoir une prédiction indiquant si l'information est vraie ou fausse.

Il est disponible sur notre dépôt GitHub : [Lien vers le dépôt GitHub](#).

Voici une capture d'écran de l'interface utilisateur :



## Fake News Detection

**Title:**

"NASA Discovers New Planet in Our Solar System"

**Text:**

"NASA scientists have announced the discovery of a new planet in our solar system. This groundbreaking discovery was made using advanced telescopes and space exploration technology. The new planet, which has been named 'Planet X', is located beyond Neptune

Predict

Prediction: False

FIGURE 13 – Interface pour la prédiction

## 6 ELI5

Pour aller plus loin, nous avons utilisé la bibliothèque ELI5 pour expliquer les décisions prises par le modèle SVC pour la première classification True vs False en utilisant 10% des données. Nous aurions voulu utiliser un plus grand volume de données, mais cela prend énormément de temps. L'objectif principal était de comprendre quels mots spécifiques dans le texte ont influencé le modèle pour classer une nouvelle comme étant vraie ou fausse.



Prediction for the sample text: False

**y=0** (probability **0.927**, score **-1.000**) top features

Contribution?	Feature
+0.524	<BIAS>
+0.085	bush
+0.062	aufhauser
+0.050	adm
+0.039	crandall
+0.037	military
+0.037	rear
+0.029	jag
+0.020	cusumano
+0.019	friday
+0.017	defense
+0.014	team
+0.014	tribunal
+0.013	judge
+0.012	patient
+0.011	george
+0.011	november
+0.011	morning
+0.010	It
+0.009	asked
+0.009	request
+0.009	evidence

FIGURE 14 – Explication des décisions du modèle SVC avec ELI5

La prédiction faite par le modèle SVC pour l'échantillon de texte est "False", ce qui signifie que le modèle a classé ce texte comme étant faux. La probabilité associée à cette prédiction est de 92,7 %, indiquant une haute confiance du modèle dans sa décision. Parmi les caractéristiques les plus influentes dans cette décision, on retrouve des mots comme "bush", "aufhauser", "adm", "crandall" et "military", chacun ayant une contribution positive significative, ce qui pousse la décision vers "False". Le terme "bush", par exemple, a une contribution de 0.085, ce qui signifie qu'il influence fortement le modèle à classer le texte comme faux. D'autres termes comme "tribunal", "judge" et "defense" jouent également un rôle important.

## 7 Conclusion

Pour conclure, ce projet nous a montré l'importance de la préparation des données et du choix des méthodes de rééquilibrage pour la classification des fake news. Bien que nous ayons obtenu des résultats satisfaisants pour les classifications binaires, les défis de la classification multiple ont mis en lumière les limitations des techniques actuelles. Une des raisons principales est l'application du downsampling. Bien que nécessaire pour équilibrer les classes, cette méthode a réduit significativement la quantité de données disponibles pour l'entraînement, impactant ainsi les performances des modèles, notamment pour la dernière classification. Le downsampling réduit le nombre de données d'entraînement, ce qui est critique lorsque le dataset initial est déjà limité.

Quelques améliorations potentielles incluent l'augmentation de la taille des datasets, ce qui pourrait aider à améliorer les performances des modèles, particulièrement pour les classifications plus complexes. L'utilisation de techniques avancées, comme les réseaux de neurones, pourrait également améliorer les performances de la classification. De plus, affiner encore plus les hyperparamètres des classifieurs et essayer différents algorithmes permettrait de trouver les configurations optimales. Enfin, utiliser des techniques d'explicabilité comme ELI5 ou SHAP pourrait aider à mieux comprendre pourquoi un modèle fait certaines prédictions et ajuster les modèles en conséquence.

Globalement, ce projet fut très intéressant et a poussé certains d'entre nous à aimer la data science et à vouloir se spécialiser dans ce domaine, malgré le fait d'être initialement orientés vers le génie logiciel.