



Rapport Technique – détection des fakes news avec l'IA



Sommaire

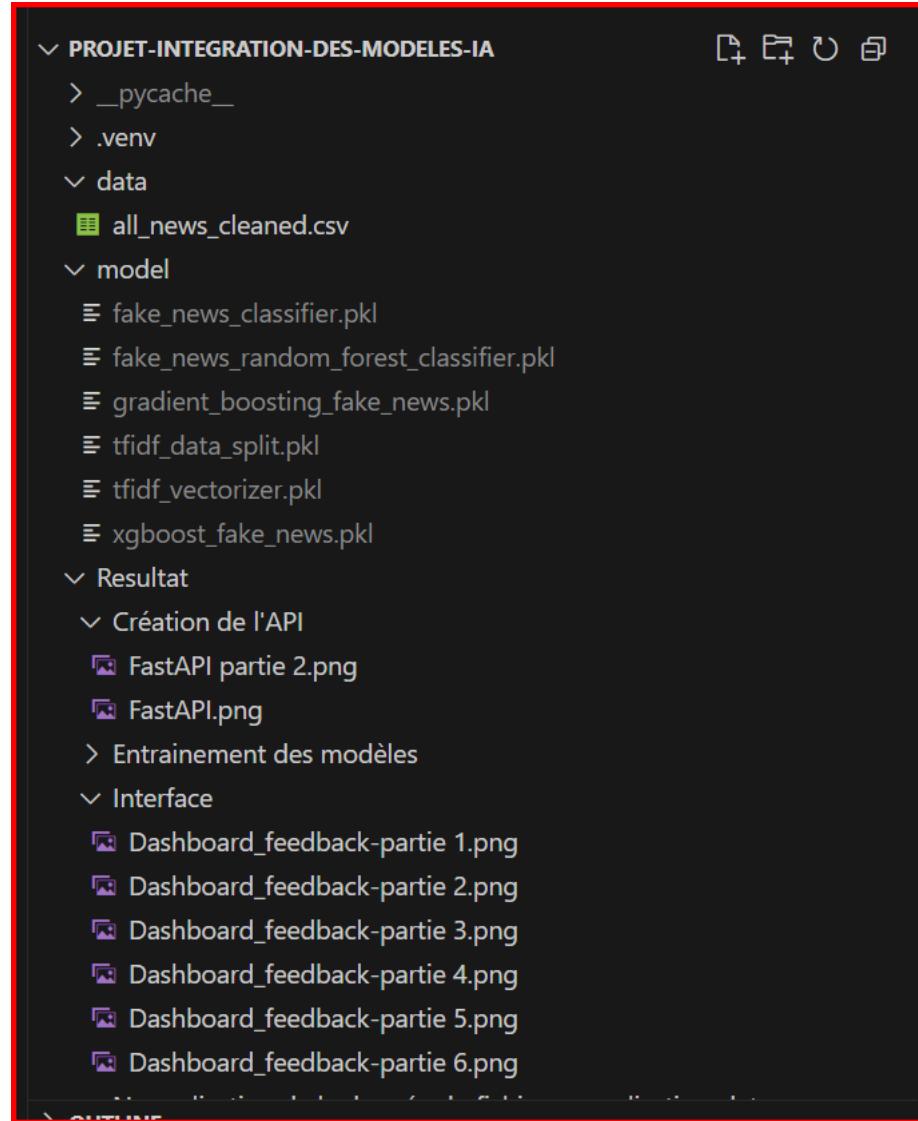
Table des matières

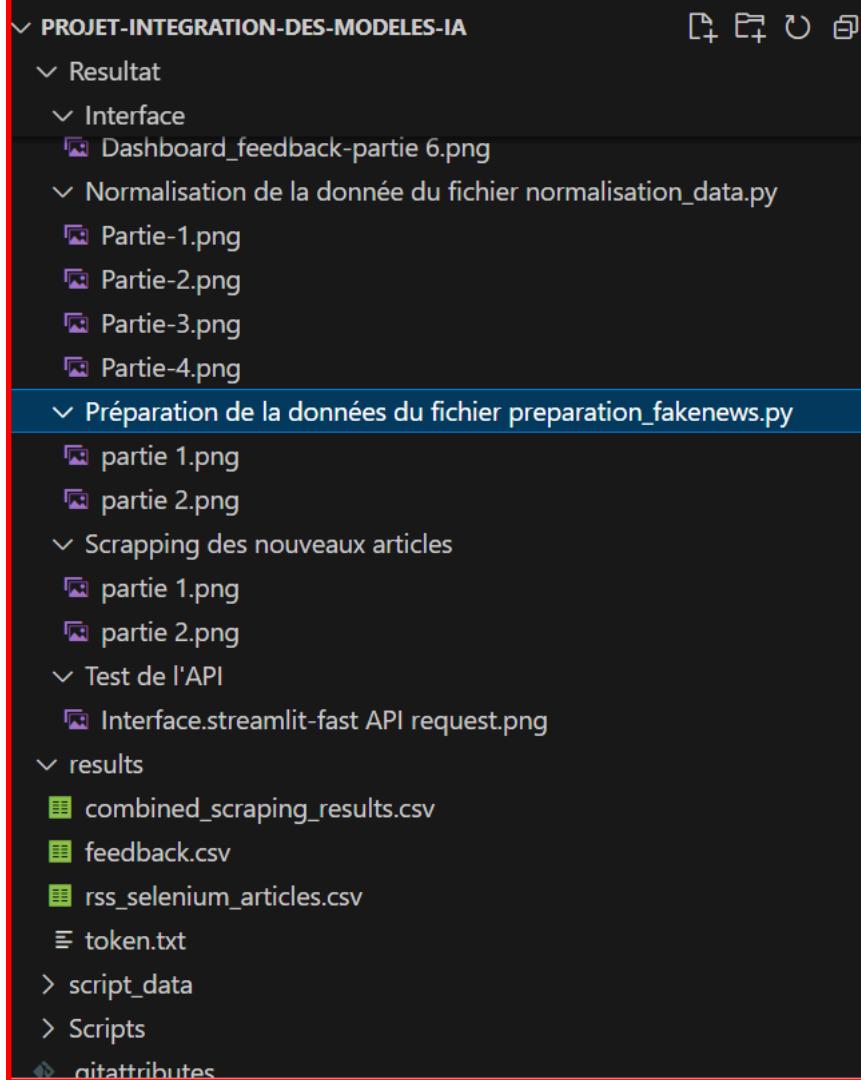
1.	Introduction	3
2.	Structure du Projet.....	3
	A- Etapes	5
	B. Instructions :	7
3.	Prétraitement et Vectorisation (Processer)	8
4.	Entraînement des Modèles (Modéliser)	9
5.	Déploiement et Interface (Robotiser).....	9
6.	Tests et Généralisation (Généraliser)	9
7.	Boucle de Feedback (Data-driven)	10
8.	Conclusion et Perspectives	11
9.	GitHub	11

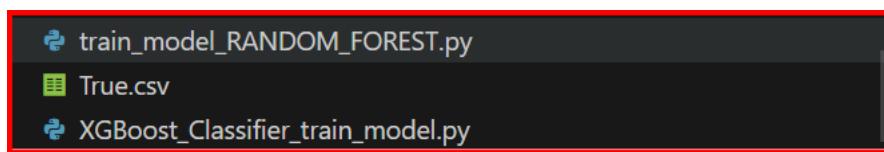
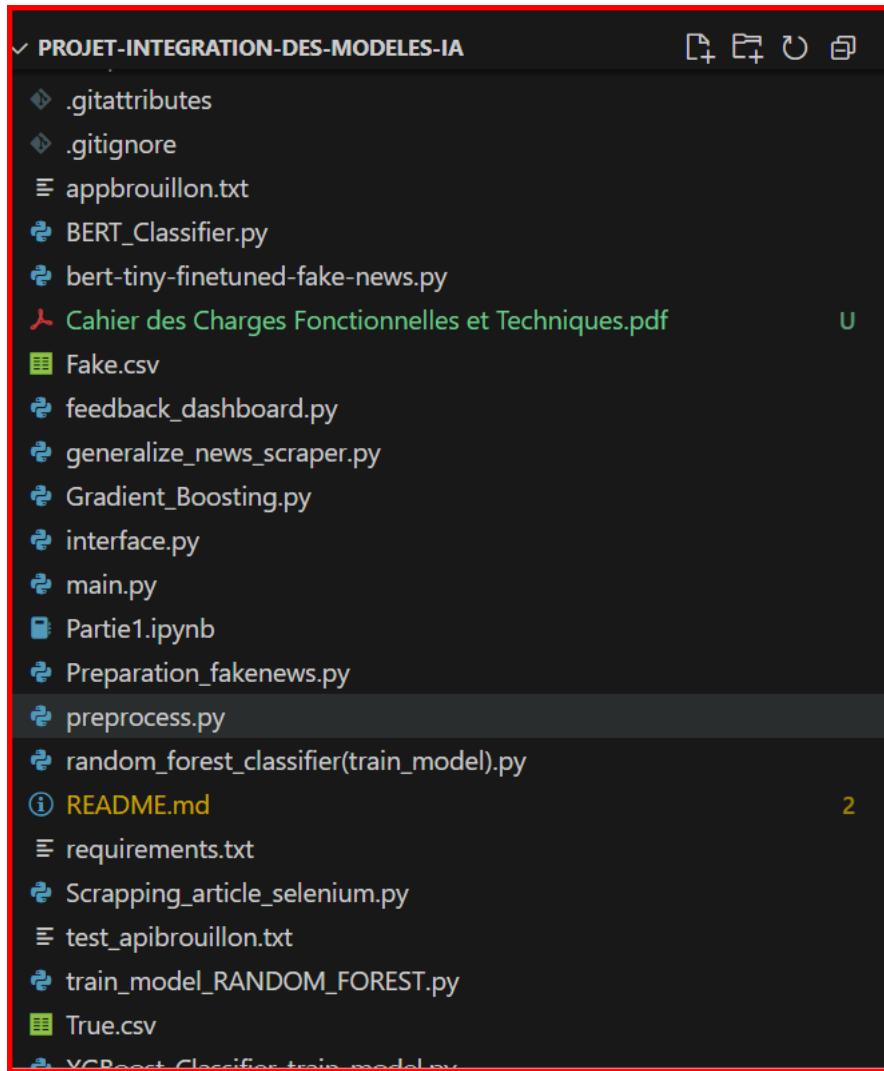
1. Introduction

Ce rapport présente la conception, l'implémentation et l'évaluation de notre solution de détection de fake news basée sur l'intelligence artificielle. Il détaille également l'organisation technique du projet, la structure des dossiers et fichiers, et les outils utilisés.

2. Structure du Projet







A- Etapes

1. Dossier Résultat :

Ce dossier contient des résultats visuels et des captures d'écran liés aux différentes étapes du projet.

2. Sous-dossiers :

1. `Création de l'API` :

- `FastAPI partie 2.png` : Capture d'écran de la deuxième partie de la création de l'API.
- `FastAPI.png` : Capture d'écran de la première partie de la création de l'API.

2. `Entrainement des modèles` :

- Contient probablement des résultats ou des visualisations liés à l'entraînement des modèles.

3. `Interface` :

- `Dashboard_feedback-partie 1.png` à `Dashboard_feedback-partie 6.png` : Captures d'écran des différentes parties du tableau de bord de feedback.

4. `Normalisation de la donnée du fichier normalisation_data.py` :

- `Partie-1.png` à `Partie-4.png` : Captures d'écran illustrant les étapes de normalisation des données.

5. `Préparation de la données du fichier preparation_fakenews.py` :

- `partie 1.png` et `partie 2.png` : Captures d'écran des étapes de préparation des données.

6. `Scrapping des nouveaux articles` :

- `partie 1.png` et `partie 2.png` : Captures d'écran des étapes de scraping des articles.

7. `Test de l'API` :

- `Interface.streamlit-fast API request.png` : Capture d'écran montrant un test de l'API via Streamlit.

Dossier model :

Ce dossier contient les modèles entraînés et les fichiers associés.

- `fake_news_classifier.pkl` : Modèle de classification des fake news.
- `fake_news_random_forest_classifier.pkl` : Modèle Random Forest pour la classification des fake news.
- `gradient_boosting_fake_news.pkl` : Modèle Gradient Boosting pour la classification des fake news.
- `tfidf_data_split.pkl` : Données transformées et divisées à l'aide de TF-IDF.
- `tfidf_vectorizer.pkl` : Vectoriseur TF-IDF utilisé pour transformer les données textuelles.
- `xgboost_fake_news.pkl` : Modèle XGBoost pour la classification des fake news.

Dossier data :

Ce dossier contient les données utilisées dans le projet.

- `all_news_cleaned.csv` : Fichier CSV contenant les données nettoyées des actualités.

Dossier results :

Ce dossier contient les résultats générés par le projet.

- `combined_scraping_results.csv` : Résultats combinés du scraping des articles.
- `feedback.csv` : Fichier contenant les feedbacks des utilisateurs.
- `rss_selenium_articles.csv` : Articles extraits via Selenium.
- `token.txt` : Fichier texte contenant probablement un jeton d'authentification ou d'accès.

Dossier script_data :

Ce dossier contient des scripts auxiliaires pour la préparation des données et l'entraînement des modèles.

- `Normalisation_data.py` : Script pour normaliser les données.
- `train_model_SMOTE.py` : Script pour entraîner un modèle avec la technique SMOTE (Synthetic Minority Oversampling Technique).
- `vectorize_data.py` : Script pour vectoriser les données textuelles.

Dossier __pycache__ :

- Contient des fichiers Python compilés automatiquement générés pour améliorer les performances.

Dossier .venv:

- Contient l'environnement virtuel Python avec toutes les dépendances installées pour le projet.

B. Instructions :

📁 Structure

- `data/` : Données nettoyées
- `model/` : Modèle entraîné et TF-IDF vectorizer
- `scripts/` : Scripts de préparation
- `interface.py` : Interface Streamlit pour la détection des fakes news et le choix des modèles de test.
- `feedback_dashboard.py` : Interface Streamlit du tableau de bord pour le système des feedbacks
- `main.py` : Fichier de l'api FastAPI

✿ Lancer l'application

```
```bash
```

```
activate ou .\.venv\Scripts\activate Commandes windows dans le terminal
powershell de vscode du projet pour activer l'environement virtuel du dossier
".venv"

pip install -r requirements.txt Pour installer toutes librairies du projet

uvicorn main:app --reload commande pour Lancer l'api FastAPI, il joue aussi le
role d'un serveur pour les deux interfaces.

http://127.0.0.1:8000/ Après avoir lancer l'API, il tourne dans le port 80,
c'est lien de l'api FastAPI

streamlit run interface.py commande pour lancer l'interface pour la détection
des fakesnews

http://localhost:8501/ Après avoir lancer l'interface de détection des fakes
news, il tourne dans port 85, c'est lien de l'interface interface.py

streamlit run feeback_dashboard.py commande pour lancer l'interface du dashboard
des feedback

http://localhost:8502/ Après avoir lancer l'interface du système des feebacks,
il tourne dans port 8502, c'est lien de l'interface feeback_dashboard.py

🔗 https://sites.google.com/chromium.org/driver/ Télécharger le web driver,
cette extension est utile pour faire le scrapping des nouveaux articles afin de
tester les modèles.
```

### 3. Prétraitement et Vectorisation (Processer)

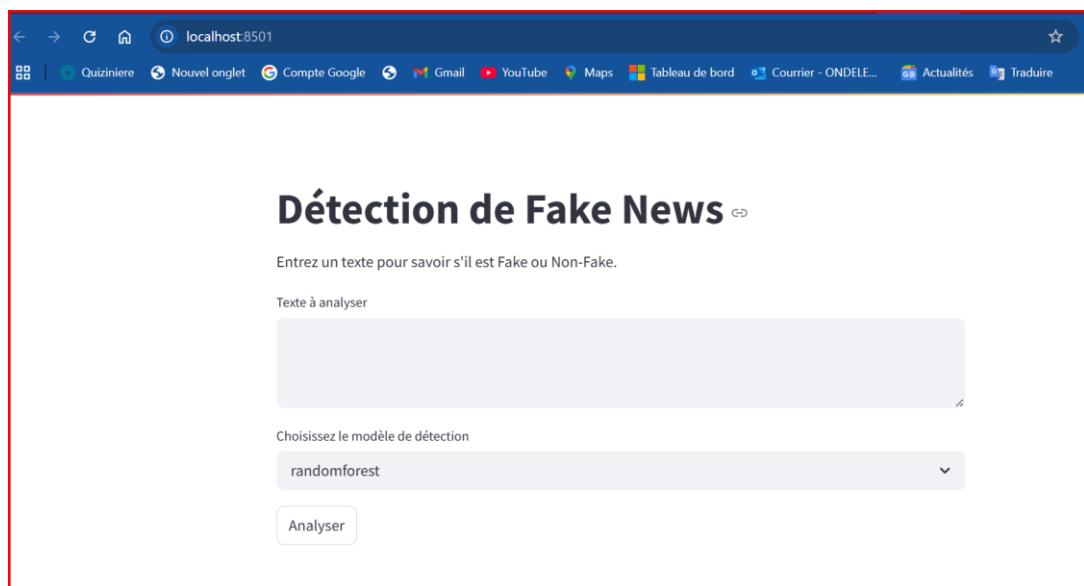
- **Nettoyage** : suppression des URLs, balises HTML, ponctuation et normalisation des caractères.
- **Fusion** : combiner fake.csv et true.csv en un seul dataset (all\_news\_cleaned.csv).
- **Vectorisation** : TF-IDF avec 10 000 features, sauvegardé dans tfidf\_vectorizer.pkl.

## 4. Entraînement des Modèles (Modéliser)

Modèle	Emplacement	F1-score
TF-IDF + Random Forest	fake_news_random_forest_classifier.pkl	~93 %
XGBoost	xgboost_fake_news.pkl	~99 %
Gradient Boosting	gradient_boosting_fake_news.pkl	~95 %
BERT (fine-tuné)	via transformers.pipeline	~96 %

## 5. Déploiement et Interface (Robotiser)

- **API FastAPI** : endpoints / (statut) et /predict (JSON in/out). Chargement des modèles joblib et transformers.
- **Streamlit** :
  - interface.py : champ de texte, choix du modèle et affichage des prédictions.



- feedback\_dashboard.py : formulaire de feedback et visualisation des statistiques.

## 6. Tests et Généralisation (Généraliser)

- **Scraping RSS + Selenium** : récupérer des articles externes (BBC, Reuters).

- **Validation cross-source** : tests sur blogs, tweets et flux divers.
- **Analyse des performances** : robustesse sur contenus courts et informels.

## 7. Boucle de Feedback (Data-driven)

- **Collecte** : formulaire utilisateur enregistrant feedback.csv.
- **Visualisation** : dashboard Streamlit mis à jour en temps réel.

**Système de Feedback - Détection de Fake News**

Envoyer un signalement

Veuillez signaler si la prédition affichée semble incorrecte.

Modèle utilisé

bert

Texte analysé

Envoyer

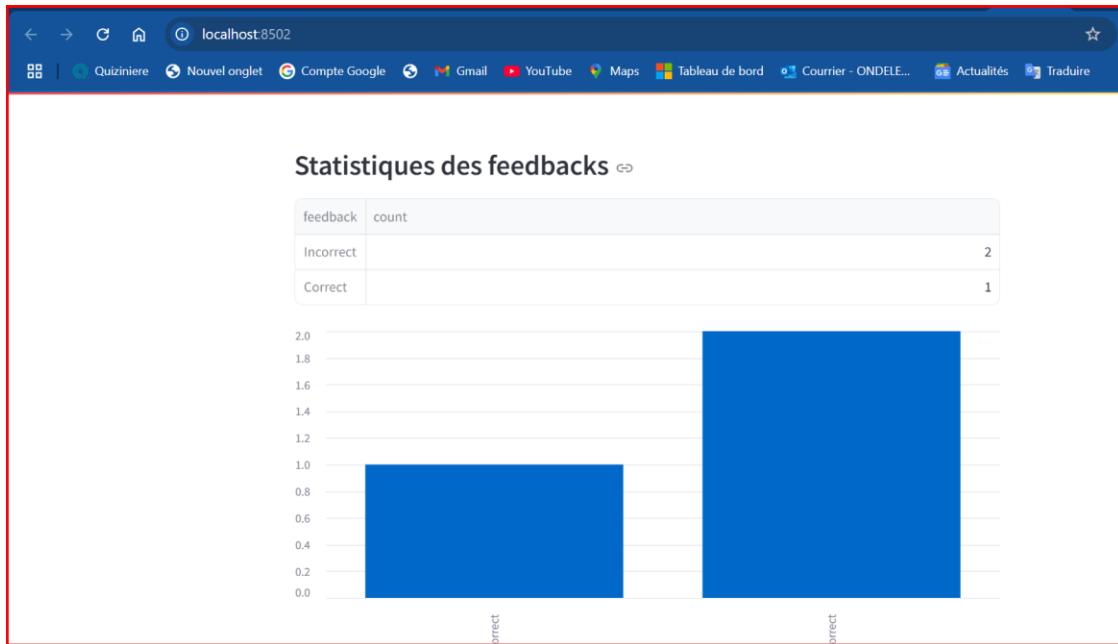
Envoyer le feedback

**Dashboard de Feedback**

**Feedbacks récents**

	timestamp	model	text	predicted	feedback	comment
0	2025-04-15T23:56:09.884297	bert	Donald trump n'est pas président	Fake	Incorrect	je n'aime
1	2025-04-16T12:07:16.500713	bert	Cristiano ronaldo est footballeur	Fake	Incorrect	Ce n'est p
2	2025-04-16T13:31:12.537655	bert	Messi est footballeur	Real	Correct	C'est vrai

**Statistiques des feedbacks**



- **Itérations** : réentraînements planifiés toutes les 2 semaines.

## 8. Conclusion et Perspectives

Notre solution démontre une détection efficace des fake news avec des scores F1 supérieurs à 93 % pour tous les modèles testés et une interface utilisateur complète. Les perspectives incluent :

- Déploiement Cloud sécurisé (AWS, GCP).
- Modèle multimodal (texte + image).
- Intégrer une API ayant accès à plus de ressources comme GPT.
- Ajustement dynamique via A/B testing et feedback continu.

## 9. GitHub

Voici le lien GitHub du projet : <https://github.com/GemimaOnde/Projet-Integration-des-modeles-IA>

Ce lien est accessible en mode public.

GemimaOndele / Projet-Integration-des-modeles-IA

Type  to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Projet-Integration-des-modeles-IA Public

main 2 Branches 0 Tags

Go to file Add file Code About

GemimaOndele UPDATE f0a5978 · 2 days ago 43 Commits

.venv	Sauvegarde temporaire de mes changements	3 days ago
Résultat	Mis à jour	3 days ago
Scripts	Résolution des conflits	last week
data	Mis à jour du projet	3 days ago
results	UPDATE	2 days ago
script_data	Mis à jour du projet	3 days ago

Readme Activity 0 stars 1 watching 0 forks

About Un projet autour de la matière intégration des modèles IA

Activity 0 stars 1 watching 0 forks

Releases No releases published Create a new release

script_data	Mis à jour du projet	3 days ago
.gitattributes	Ajout de .gitattributes pour gérer les fins de ligne	last week
.gitignore	Sauvegarde temporaire de mes changements	3 days ago
BERT_Classifier.py	Mis à jour du projet	3 days ago
Fake.csv	Résolution des conflits	last week
Gradient_Boosting.py	Mis à jour du projet	3 days ago
Partie1.ipynb	Mis à jour du projet	3 days ago
Preparation_fakenews.py	Mis à jour du projet	3 days ago
README.md	Mis à jour du README.md	2 days ago
Scrappling_article_selenium.py	Ajout des fichiers non suivis	3 days ago
True.csv	Résolution des conflits	last week
XGBoost_Classifier_train_model.py	Mis à jour du projet	3 days ago
appbrouillon.txt	Mis à jour du projet	3 days ago

XGBoost_Classifier_train_model.py	Mis à jour du projet	3 days ago
appbrouillon.txt	Mis à jour du projet	3 days ago
bert-tiny-finetuned-fake-news.py	Mis à jour du projet	3 days ago
feedback_dashboard.py	Mis à jour du gradient boosting	2 days ago
generalize_news_scraping.py	Ajout des fichiers non suivis	3 days ago
interface.py	UPDATE	2 days ago
main.py	Mis à jour	2 days ago
preprocess.py	Mis à jour du projet	3 days ago
random_forest_classifier(train_model).py	Mis à jour du projet	3 days ago
requirements.txt	Mis à jour du projet	3 days ago
test_apibrouillon.txt	Mis à jour du projet	3 days ago
train_model_RANDOM_FOREST.py	Mis à jour du projet	3 days ago

Suggested workflows Based on your tech stack

**SLSA Generic generator** Configure Generate SLSA3 provenance for your existing release workflows

**Pylint** Configure Lint a Python application with pylint.

**Python Package using Anaconda** Configure Create and test a Python package on multiple Python versions using Anaconda for package management.

More workflows Dismiss suggestions

Le fichier **REAME.md** qui décrit les étapes du projet et les instruction pour lancer le projet.

The screenshot shows a portion of a README.md file. At the top, there's a header section with a title and a small icon. Below it, there's a section titled "Structure" with a folder icon, followed by a bulleted list of project files. At the bottom, there's a section titled "Lancer l'application" with a gear icon.

Projet-Intégration-des-modèles-IA

## Fake News Detector

Ce projet permet de détecter si une news est vraie ou fausse à l'aide d'un modèle IA entraîné sur un corpus nettoyé de fausses et vraies actualités.

### Structure

- `data/` : Données nettoyées
- `model/` : Modèle entraîné et TF-IDF vectorizer
- `scripts/` : Scripts de préparation
- `interface.py` : Interface Streamlit pour la détection des fakes news et le choix des modèles de test.
- `feedback_dashboard.py` : Interface Streamlit du tableau de board pour le systèmes des feedbacks
- `main.py` : Fichier de l'api FastAPI

### Lancer l'application

## Lancer l'application

```
activate ou .\venv\Scripts\activate #Commandes windows dans le terminal powershell de vscode du proj
pip install -r requirements.txt #Pour installer toutes librairies du projet
uvicorn main:app --reload #commande pour Lancer l'api FastAPI, il joue aussi le role d'un serveur po
http://127.0.0.1:8000/ #Après avoir lancer l'API, il tourne dans le port 80, c'est lien de l'api Fast
streamlit run interface.py #commande pour lancer l'interface pour la détection des fakesnews
http://localhost:8501/ #Après avoir lancer l'interface de détection des fakes news, il tourne dans po
streamlit run feeback_dashboard.py #commande pour lancer l'interface du dashboard des feedback
http://localhost:8502/ #Après avoir lancer l'interface du système des feebacks, il tourne dans port 8

🔗 https://sites.google.com/chromium.org/driver/ #Télécharger le web driver, cette extension est uti
```

Le fichier **requirements.txt** qui enregistre toutes librairies, bibliothèques utilisés à installer (utiliser la commande **pip install -r requirements.txt**) pour tester le projet.

.

```
≡ requirements.txt
1 Flask==2.0.3
2 joblib==1.1.0
3 scikit-learn==1.0.2
4 requests==2.27.1
5 pandas==2.2.3
6 nltk==3.9.1
7 regex==2024.11.6
8 imbalanced-learn
9 XGBoost
10 xgboost-cpu
11 torch
12 huggingface_hub
13 hf_xet
14 transformers
15 tqdm
16 fastapi
17 uvicorn
18 numpy
19
```

---

Ce rapport a été rédigé par **Hubert CHAVASSE**, **Hadrien LENNON**, **Gémima ONDELE**, **Mohamed GHARMAOUI** et **Niangoran BOKA** dans le cadre du module **d'intégration des modèles IA**.