Note Explicative : Les Opérations Réalisées dans le Code

Ce document explique de manière précise les différentes opérations effectuées dans le code de classification de SMS en spam ou non-spam. Chaque étape est expliquée avec des détails sur son rôle, son exécution et son impact sur le pipeline global.

1. Importation des Bibliothèques

Le code commence par importer les bibliothèques nécessaires pour le traitement, l'analyse et la modélisation des données. Voici un détail des bibliothèques utilisées :

- os : Fournit des outils pour interagir avec le système de fichiers et gérer les chemins, comme la création et la vérification de dossiers.
- pandas : Une bibliothèque pour manipuler des données tabulaires. Elle est utilisée ici pour charger et préparer les données issues de fichiers CSV.
- joblib : Sert à sauvegarder et charger efficacement les objets Python, en particulier les modèles d'apprentissage automatique.
- scikit-learn : Fournit un ensemble complet d'outils pour le prétraitement des données, la modélisation (Random Forest) et l'évaluation (ROC, matrice de confusion, etc.).
- matplotlib / seaborn : Utilisées pour créer des visualisations graphiques, telles que des courbes ROC et des matrices de confusion.
- nltk : Permet le prétraitement linguistique comme la suppression des mots vides (stopwords) et la lemmatisation.
- streamlit : Une bibliothèque pour créer une interface utilisateur interactive, permettant de télécharger des fichiers et de prédire des résultats directement.

2. Déclaration des Constantes

- OUTPUT_DIR : Indique le dossier de sortie où seront enregistrés les fichiers de sauvegarde (modèle, vectoriseur, visualisations).
- MODEL_PATH/VECTORIZER_PATH : Spécifient les chemins exacts pour sauvegarder ou recharger le modèle d'apprentissage et le vectoriseur TF-IDF.

Ces chemins permettent une organisation claire des fichiers et assurent une reprise facile du travail.

3. Nettoyage des Données (Fonction `clean_text`)

Cette fonction est essentielle pour transformer les messages bruts en un format compréhensible par le modèle :
1. Conversion en Minuscules : Uniformise les mots, évitant que "Spam" et "spam" soient traités différemment.
2. Remplacement des URL : Les liens Internet sont remplacés par le mot "url" pour réduire le bruit.
3. Suppression des Caractères Non Alphabétiques : Simplifie les messages en supprimant les chiffres et les symboles inutiles.
4. Suppression des Stopwords : Élimine les mots comme "the", "is" qui n'ajoutent pas de sens contextuel.
5. Lemmatisation : Réduit les mots à leur forme de base (exemple : "running" devient "run"), rendant les textes plus homogènes.

4. Chargement et Sauvegarde du Modèle (Fonctions `load_model_and_vectorizer` et `save_model_and_vectorizer`)

- load_model_and_vectorizer :
    - Tente de recharger le modèle et le vectoriseur sauvegardés.
    - Affiche un message d'erreur via Streamlit si les fichiers n'existent pas ou sont corrompus.
- save_model_and_vectorizer :
    - Sauvegarde le modèle et le vectoriseur après l'entraînement.
    - Utilise `joblib` pour une sauvegarde rapide et efficace.

5. Entraînement du Modèle (Fonction `train_model`)

Cette fonction regroupe toutes les étapes clés pour construire un modèle performant :

1. Vectorisation des Textes :
  - Le vectoriseur TF-IDF convertit les textes en matrices numériques, où chaque mot est pondéré par sa fréquence relative.

2. Division des Données :
  - Les données sont divisées en ensembles d'entraînement (80%) et de test (20%) pour évaluer la performance sur des données inconnues.

3. Gestion du Déséquilibre des Classes :
  - Des poids adaptés sont attribués aux classes pour pallier au déséquilibre naturel entre "ham" et "spam".

4. Recherche d'Hyperparamètres :
  - GridSearchCV explore différents paramètres (comme le nombre d'arbres dans la forêt) pour optimiser les performances.

5. Visualisations :
  - Trace un graphique des résultats de GridSearch pour montrer l'impact des paramètres.
  - Génère une courbe ROC pour évaluer la capacité du modèle à différencier les classes.

6. Évaluation du Modèle (Fonction `generate_metrics`)

Pour juger la performance, cette fonction produit :

1. Rapport de Classification :
   - Fournit des mesures telles que la précision, le rappel et le score F1 pour chaque classe.

2. Matrice de Confusion :
   - Montre les prédictions correctes et les erreurs, classées par "spam" et "ham".

3. Graphiques :
   - Une matrice de confusion est affichée sous forme de carte thermique pour une interprétation visuelle.

7. Interface Utilisateur (Streamlit)

L'application Streamlit rend l'outil accessible aux utilisateurs non techniques :

1. Réentraînement du Modèle :
   - Les utilisateurs peuvent télécharger un fichier CSV contenant des SMS annotés.
   - Un nouveau modèle est entraîné sur ces données.
   - Les résultats, comme les scores F1 et la matrice de confusion, sont affichés.

2. Prédiction sur un Nouveau SMS :
   - L'utilisateur entre un message texte. - Le modèle prédit si le message est un spam ou non.
   - Les résultats sont présentés de manière claire et intuitive.

Conclusion

Ce code implémente un pipeline complet de classification des SMS, allant du prétraitement des textes à la prédiction via une interface utilisateur. Les opérations sont choisies pour offrir une solution précise et flexible. Les visualisations et l'interface utilisateur améliorent l'accessibilité et la compréhension du modèle, rendant cet outil utile pour une large variété d'utilisateurs.