Rapport d'évaluation

Table des matières

1. **Matrice de confusion**

a. **Les prédictions positives - interprétation**

```
(.env) PS C:\Users\UltraBook 3.1\OneDrive - Efrei\Documents\M2\Algo\SocialMetric_AI> python positive_training.py
Rapport de classification :
              precision    recall  f1-score   support

           0       0.90      0.69      0.78        13
           1       0.67      0.89      0.76         9

    accuracy                           0.77        22
   macro avg       0.78      0.79      0.77        22
weighted avg       0.80      0.77      0.77        22

Matrice de confusion :
[[9 4]
 [1 8]]
```

- 9 vrais négatifs : le modèle a correctement classé 9 commentaires négatifs
- 8 vrais positifs : le modèle a correctement classé 8 commentaires positifs
- 4 faux positifs : le modèle a classé 4 commentaires négatifs comme positifs
- 1 faux négatifs : le modèle a classé 1 commentaire positif comme négatifs

Calcul du raciaux

Vrai négatif = 9/13*100 = 67%
Vrai positif = 8/9*100 = 88

Conclusion

Le modèle semble bien identifier les commentaires positifs (TP = 8, FN=1). Il a un taux d'erreur plus élevé sur les négatifs (FP =4), donc il confond certains commentaires négatifs avec des positifs.

b. **Les prédictions négatives – interprétation**

```
(.env) PS C:\Users\UltraBook 3.1\OneDrive - Efrei\Documents\M2\Algo\SocialMetric_AI> python negative_training.py
Rapport de classification :
              precision    recall  f1-score   support

           0       0.67      0.89      0.76         9
           1       0.90      0.69      0.78        13

    accuracy                           0.77        22
   macro avg       0.78      0.79      0.77        22
weighted avg       0.80      0.77      0.77        22

Matrice de confusion :
[[8 1]
 [4 9]]
```

- 8 vrais négatifs : le modèle a classé 8 commentaires négatifs
- 9 vrais positifs : le modèle a classé 9 commentaires positifs
- 1 faux positif : le modèle a classé 1 commentaire négatif comme positif

- 4 faux négatifs : le modèle a classé 4 commentaires positifs comme négatif

Calcul du raciaux

Vrai négatif = 8/9*100 = 88%
Vrai positif = 4/13*100 = 30%

Conclusion

Le modèle semble bien identifier les commentaires positifs (TP = 4, FN = 4). Il a un taux d'erreur moins élevé sur les négatifs (FP = 1).

2. **Calcul et interprétation des mesures**

a. **Cas positif**

Rapport de classification :

	precision	recall	f1-score	support
0	0.90	0.69	0.78	13
1	0.67	0.89	0.76	9

- **Précision**

Pour les négatifs (0) : 0.90 ⍰ lorsque le modèle prédit un commentaire négatif, il est correct 90%. Pour les positifs (1) : 0.67 ⍰ lorsque le modèle prédit un commentaire positif, il est correct à 67%.

- **Recall**

Pour les négatives (0) : 0.69 ⍰ 69% des commentaires négatifs sont bien détectés.
Pour les positifs (1) : 0.89% ⍰ 89% des commentaires positifs sont bien détectés.

- **F1-score**

Sachant que le f1-score du cas négatifs est de 0.78 et celui du cas positif est 0.76, on peut dire que le model est performant. Le model détecte mieux les cas négatifs que les cas négatifs.

b. **Cas négatif**

	precision	recall	f1-score	support
0	0.67	0.89	0.76	9
1	0.90	0.69	0.78	13

- **Précision**

0(négatif) : 0.67 ⇒ 67% des prédictions négatives sont
correctes 1 (positif) : 0.90 ⇒ 90% des prédictions positives
sont correctes

- **Recall**

Pour les négatives (0) : 0.89 ⇒ 89% des vrais commentaires négatifs sont bien
détectés Pour les positifs (1) : 0.69% ⇒ 69% des vrais commentaires positifs sont
bien détectés

- **F1-score**

Sachant que le f1-score du cas négatifs est de 0.76 et celui du cas positif est 0.78, on peut dire que le model est performant. Le model détecte mieux les cas positifs que les cas négatifs

3. Analyses des performances

a. Forces

Bonne précision pour les commentaires négatifs (0.90)

- ⇒ Lorsqu'il prédit un commentaire comme négatif, il a 90% de chances d'avoir raison.
- ⇒ Cela signifie qu'il fait peu d'erreurs en attribuant des commentaires négatifs.
- ⇒ Il serait donc plus facile dans le système de détecter les commentaires négatifs et de les flouter par exemples.

Bon recall pour les commentaires positifs (0.89)

- ⇒ 89% des vrais commentaires positifs sont bien détectés.
- ⇒ Ceci signifie que le modèle capture presque tous les commentaires positifs, ce qui est utile si l'on veut éviter de manquer des retours positifs.

Bon équilibre entre précision et recall (F1-score sensiblement égale à 0.77)

- ⇒ Avec 0.78 pour les positifs et 0.76 pour les négatifs, le modèle montre la performance globale.
- ⇒ Il est capable de détecter à la fois les commentaires négatifs et positifs sans un biais trop fort en faveur de l'un ou de l'autre.

b. Faiblesses

Mauvaise précision pour les commentaires positifs

- ⇒ Lorsqu'il prédit un commentaire comme positif, il se trompe dans 33% des cas.
- ⇒ Il classe certains commentaires négatifs à tort comme positifs (4 faux positifs).
- ⇒ Peut poser problème si l'on veut éviter de promouvoir des avis négatifs.

Faible recall pour les commentaires négatif (0.69)

- ⇒ 31% des commentaires négatifs ne sont pas détectés et sont classés comme positifs.
- ⇒ Si le but est d'éviter les retours négatifs, ce recall n'est pas optimal.

Tendances à sous-estimer les commentaires négatifs

- Le modèle classifie trop de négatifs comme positifs (FP=4)
- Si le but est de filtrer correctement les commentaires négatifs, cette faiblesse peut l'affecter.

c. Biais

Influence du langage et du contexte

- Ironie et sarcasme
  Exemple : "Super service ! J'ai attendu 2 heures pour une réponse…" Ce type de commentaire peut être mal classé comme positif.
- Langage informel, emojis et variations linguistiques peuvent aussi poser problème.

Biais des données

- La distribution des commentaires dans l'ensemble d'apprentissage peut influencer les performances.
- Si les commentaires positifs sont sous-représentés, le modèle peut être moins performant pour les identifier correctement.

Biais dans le seuil de décision

- Un ajustement du seuil de classification pourrait améliorer la détection des commentaires négatifs ou positifs selon l'objectif.
- Baisser le seuil pour les positifs améliorerait le rappel mais pourrait augmenter les faux positifs.
- Augmenter le seuil pour les négatifs pourrait aider à mieux détecter les retours négatifs.

4. Recommandations claires pour améliorer le modèle Amélioration des données d'entraînement

- ✓ **Équilibrer le dataset**

  Vérifier si le dataset contient autant de tweets positifs que négatifs. Si un déséquilibre existe, utiliser des techniques comme :

  - Oversampling (ajouter des exemples de la classe sous-représentée).
  - Undersampling (réduire le nombre d'exemples de la classe dominante).

- ✓ **Améliorer la diversité des données**

  Ajouter des tweets contenant du sarcasme, des emojis et des expressions informelles pour mieux capter le contexte linguistique.

Vérifier que les différents styles d'écriture (langage formel/informel, fautes d'orthographe) sont bien représentés.

- ✓ **Nettoyage et prétraitement avancé**

  Tester différentes méthodes de tokenization (ex. WordPiece, Byte-Pair Encoding) pour mieux représenter le texte.
  Ajouter la prise en compte des emojis et du contexte (ex. 😭 souvent associé à un tweet négatif).

Optimisation du modèle

- ✓ **Tester un modèle NLP plus avancé**

  - Remplacer la régression logistique par un modèle basé sur du deep learning comme :
    - BERT (Bidirectional Encoder Representations from Transformers)
    - RoBERTa
    - XLNet

  - Ces modèles comprennent mieux le contexte des phrases et améliorent la détection du sarcasme et des ambiguïtés.

- ✓ **Ajuster le seuil de classification**

  - Actuellement, le modèle produit trop de faux positifs (4).
  - Tester différents seuils (ex. 0.6 au lieu de 0.5) pour réduire les erreurs de classification des tweets négatifs en positifs.

- ✓ **Expérimenter avec d'autres techniques de pondération**

  - Tester des métriques alternatives comme le balanced accuracy ou focal loss pour mieux gérer le déséquilibre des classes.

5. **Planification du réentraînement du modèle**

a. **Fonctionnement du script retrain_model.py**

Le script retrain_model.py est conçu pour réentraîner le modèle de régression logistique en utilisant des données de tweets stockées dans la base de données MySQL.

Voici les étapes principales du script :

(1) **Importation des bibliothèques nécessaires :**

```
import sqlalchemy
import pymysql
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
import logging
```

(2) **Configuration de la journalisation :**

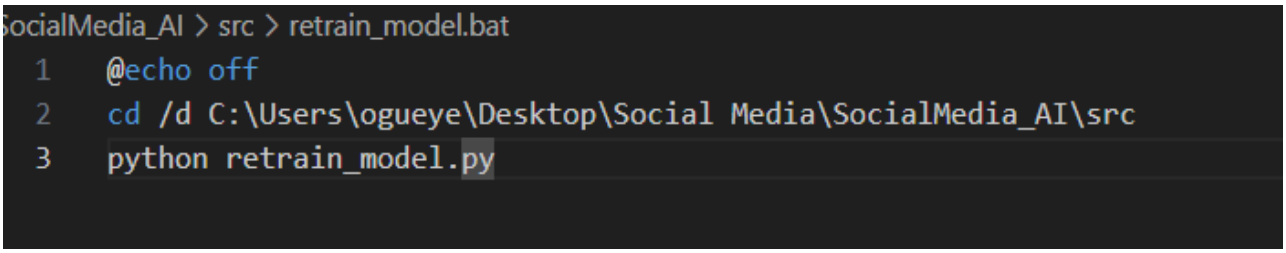
```
# Configurer la journalisation
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(message)s')
```

(3) **Définition de la fonction retrain_model**

```python
def retrain_model():
    logging.info("Début du réentrainement du modèle")
    try:
        #  Connexion à la base de données MySQL
        logging.info("Connexion à la base de données MySQL...")
        engine = sqlalchemy.create_engine('mysql+pymysql://user:userpassword@localhost/tweets_db')
        logging.info("Connexion réussie à la base de données MySQL")

        query = "SELECT text, positive, negative FROM tweets"
        df = pd.read_sql(query, engine)
        logging.info("Données récupérées depuis la base de données MySQL")
        logging.info(f"Nombre de lignes récupérées : {len(df)}")

        #  Fonction de nettoyage du texte
        def clean_text(text):···

        df['text_clean'] = df['text'].apply(clean_text)
        logging.info("Nettoyage du texte terminé")

        # Vérifier la distribution des classes avant la séparation des données
        logging.info(f"Distribution des classes (positif) avant la séparation : {df['positive'].value_counts().to_dict()}")
        logging.info(f"Distribution des classes (négatif) avant la séparation : {df['negative'].value_counts().to_dict()}")

        #  Stopwords français (à filtrer dans la vectorisation)
        french_stopwords = [···

        vectorizer = CountVectorizer(stop_words=french_stopwords, max_features=100)
        X = vectorizer.fit_transform(df['text_clean'])
        y_positive = df['positive']
        y_negative = df['negative']

        #  Séparation des données en train/test
        X_train_pos, X_test_pos, y_train_positive, y_test_positive = train_test_split(X, y_positive, test_size=0.25, random_state=42)
        X_train_neg, X_test_neg, y_train_negative, y_test_negative = train_test_split(X, y_negative, test_size=0.25, random_state=42)

        # Vérifier la distribution des classes après la séparation des données
        logging.info(f"Distribution des classes dans y_train_positive : {pd.Series(y_train_positive).value_counts().to_dict()}")
        logging.info(f"Distribution des classes dans y_train_negative : {pd.Series(y_train_negative).value_counts().to_dict()}")

        # Vérifier si les classes sont équilibrées
        if len(y_train_positive.unique()) < 2 or len(y_train_negative.unique()) < 2:···

        #  Entrainement du modèle de régression logistique pour les tweets positifs
        model_positive = LogisticRegression()
        model_positive.fit(X_train_pos, y_train_positive)

        #  Entrainement du modèle de régression logistique pour les tweets négatifs
        model_negative = LogisticRegression()
        model_negative.fit(X_train_neg, y_train_negative)

        #  Prédictions et évaluation du modèle pour les tweets positifs
        y_pred_positive = model_positive.predict(X_test_pos)
        logging.info("Rapport de classification (positif) :")
        logging.info("\n" + classification_report(y_test_positive, y_pred_positive))

        logging.info("Matrice de confusion (positif) :")
        logging.info("\n" + str(confusion_matrix(y_test_positive, y_pred_positive)))

        #  Prédictions et évaluation du modèle pour les tweets négatifs
        y_pred_negative = model_negative.predict(X_test_neg)
        logging.info("Rapport de classification (négatif) :")
        logging.info("\n" + classification_report(y_test_negative, y_pred_negative))

        logging.info("Matrice de confusion (négatif) :")
        logging.info("\n" + str(confusion_matrix(y_test_negative, y_pred_negative)))

    except Exception as e:
        logging.error("Erreur lors du réentrainement du modèle : %s", e)

if __name__ == "__main__":
    retrain_model()
```

b. Fonctionnement du fichier batch retrain_model.bat

Le fichier batch retrain_model.bat est utilisé pour exécuter le script Python retrain_model.py via le Planificateur de tâches de Windows.
Voici son contenu :

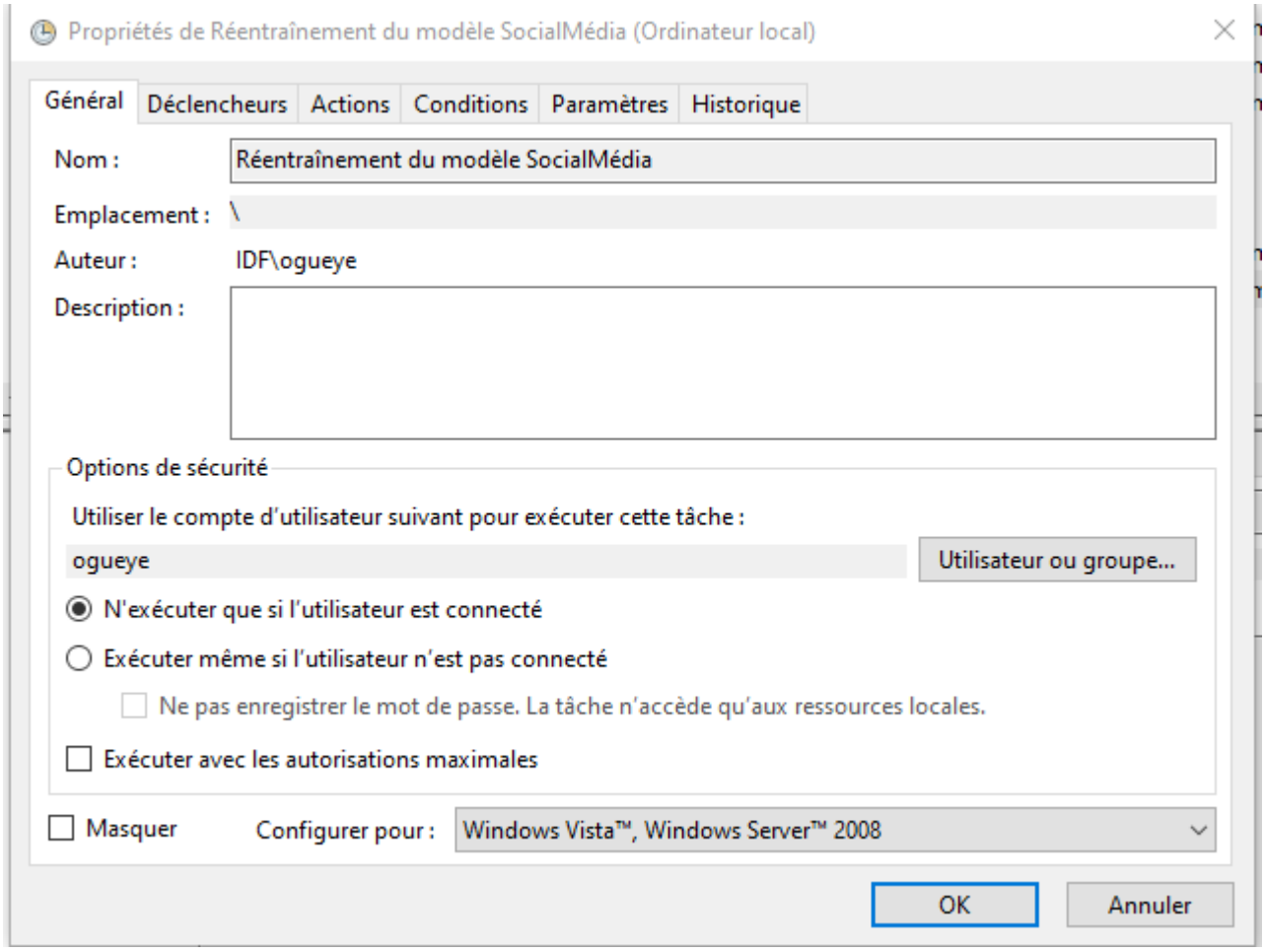


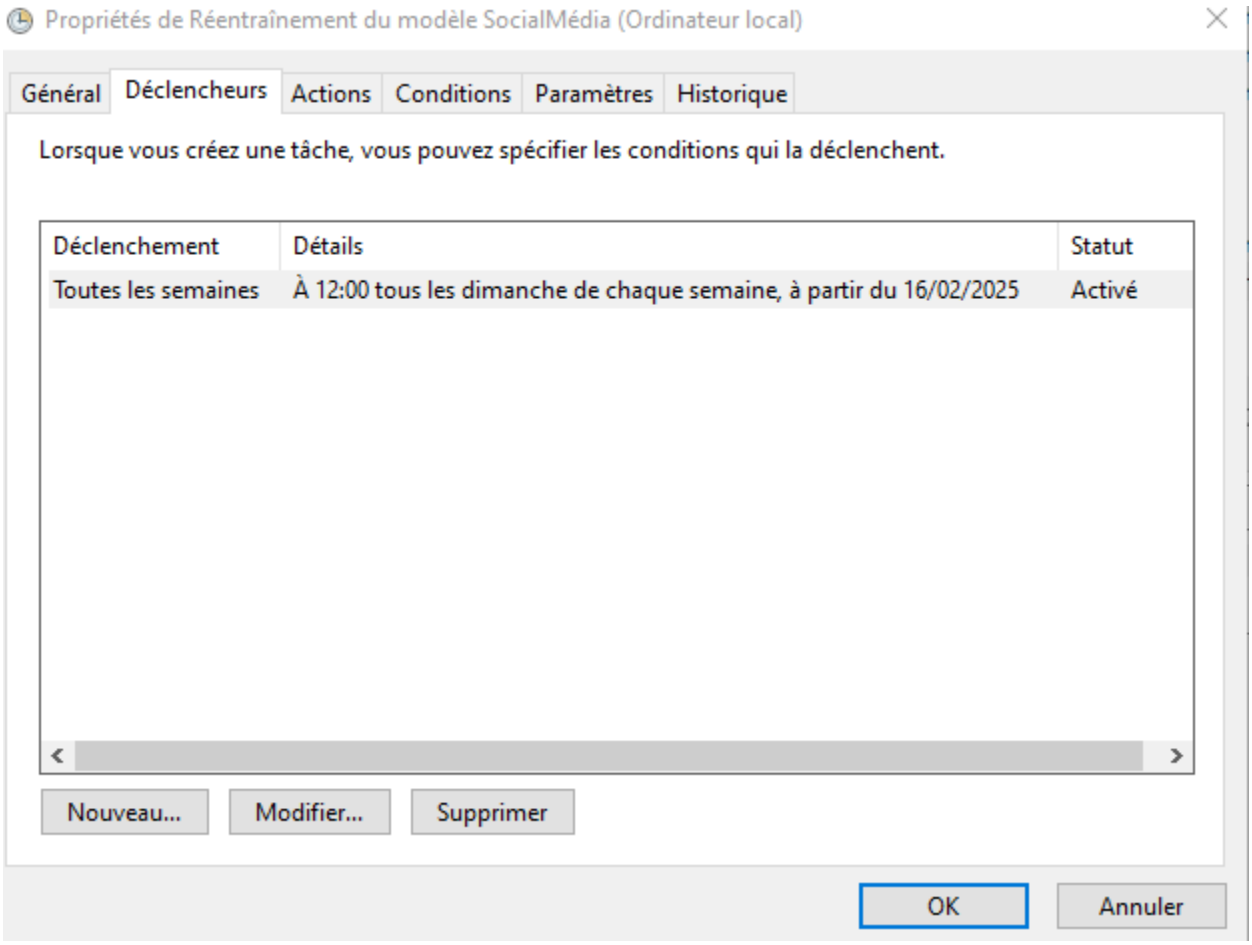
Explication du fichier batch

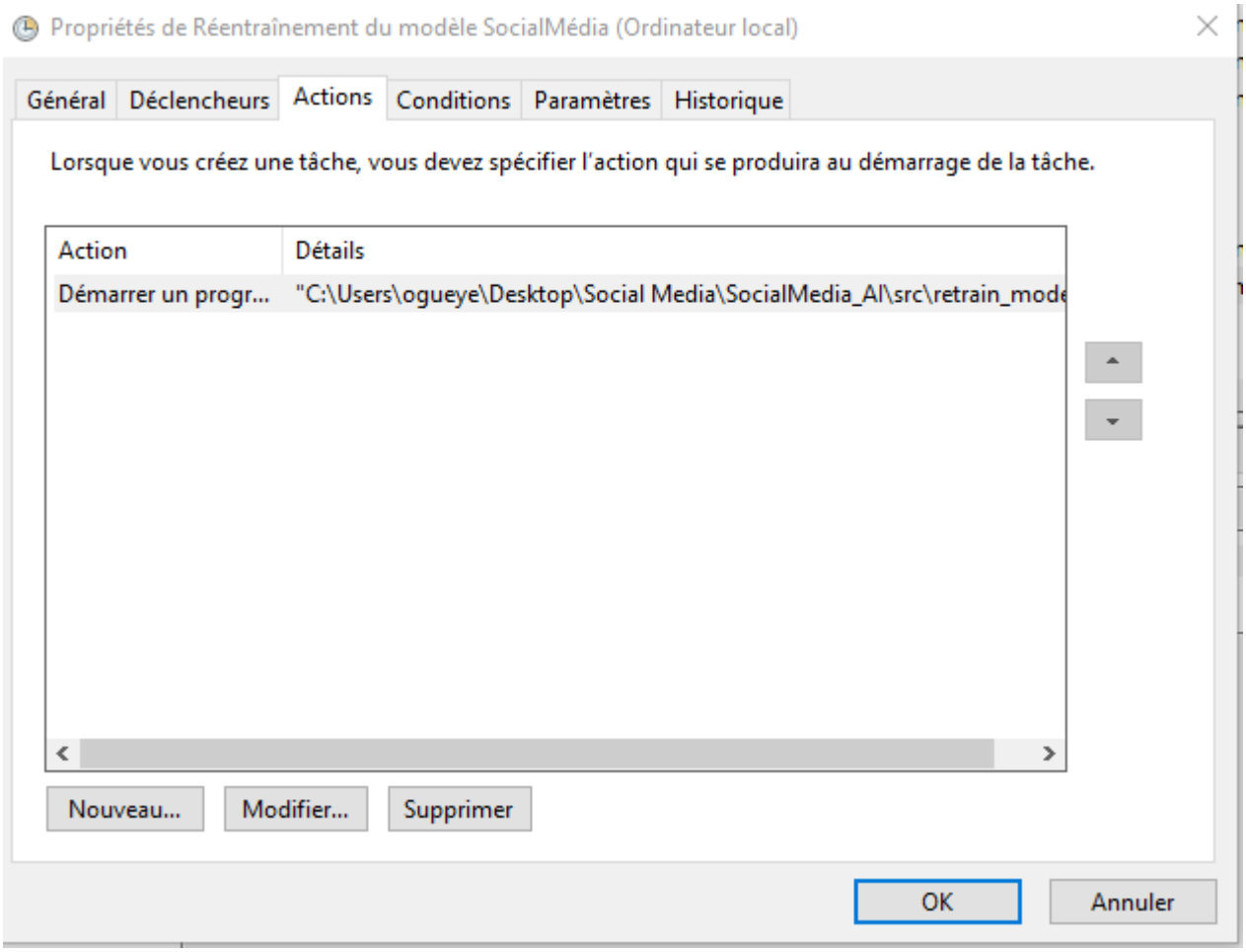
1. @echo off : Cette commande désactive l'affichage des commandes dans la fenêtre de commande.
2. cd /d C:\Users\ogueye\Desktop\Social Media\SocialMedia_AI\src : Cette commande change le répertoire de travail actuel pour le répertoire où se trouve le script Python.
3. python retrain_model.py : Cette commande exécute le script Python retrain_model.py.

c. Planification du réentraînement du modèle

Pour automatiser le réentraînement du modèle, on ouvre le Planificateur de tâches de Windows pour exécuter le fichier batch retrain_model.bat tout les dimanches à 12h00mn.
Voici les étapes pour configurer le Planificateur de tâches :


Propriétés de Réentraînement du modèle SocialMédia (Ordinateur local)
Général
Déclencheurs
Actions
Conditions
Paramètres
Historique
Nom :
Réentraînement du modèle SocialMédia
Emplacement :
\
Auteur :
IDF\ogueye
Description :
Options de sécurité
Utiliser le compte d'utilisateur suivant pour exécuter cette tâche :
ogueye
Utilisateur ou groupe...
N'exécuter que si l'utilisateur est connecté
Exécuter même si l'utilisateur n'est pas connecté
Ne pas enregistrer le mot de passe. La tâche n'accède qu'aux ressources locales.
Exécuter avec les autorisations maximales
Masquer
Configurer pour :
Windows Vista™, Windows Server™ 2008
OK
Annuler


Propriétés de Réentraînement du modèle SocialMédia (Ordinateur local)
Général
Déclencheurs
Actions
Conditions
Paramètres
Historique
Lorsque vous créez une tâche, vous pouvez spécifier les conditions qui la déclenchent.
Déclenchement
Détails
Statut
Toutes les semaines
À 12:00 tous les dimanche de chaque semaine, à partir du 16/02/2025
Activé
Nouveau...
Modifier...
Supprimer
OK
Annuler



Avec cette configuration, le réentraînement du modèle sera automatisé et exécuté àtous les dimanches à 12h00, garantissant ainsi que le modèle reste à jour avec les nouvelles données.