In [3]:

```python
def recommandation_finale(tconst):

    import pandas as pd
    from sklearn.neighbors import NearestNeighbors
    from sklearn.preprocessing import MinMaxScaler

    # 1ere reco : 5 films avec genre commun et pays commun

    # Chargement des données
    df_ml = pd.read_csv("../machine learning/DF_ML.csv.gz")

    # Récupération des valeurs genre et pays qui correspondent au film sélection
    df_selection = df_ml[df_ml['tconst'] == tconst]
    colonnes_genre = [
        'Action', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime',
        'Documentary', 'Drama', 'Family', 'Fantasy', 'Game-Show', 'History',
        'Horror', 'Music', 'Musical', 'Mystery', 'News', 'Reality-TV',
        'Romance', 'Sci-Fi', 'Sport', 'Talk-Show', 'Thriller', 'War', 'Western'
    ]
    colonnes_pays = [
        'tmdb_US', 'tmdb_FR', 'tmdb_GB', 'tmdb_DE', 'tmdb_JP', 'tmdb_IN',
        'tmdb_IT', 'tmdb_CA', 'tmdb_ES', 'tmdb_MX', 'tmdb_HK', 'tmdb_BR',
        'tmdb_SE', 'tmdb_SU', 'tmdb_PH', 'tmdb_KR', 'tmdb_AU', 'tmdb_CN',
        'tmdb_AR', 'tmdb_RU', 'tmdb_DK', 'tmdb_NL', 'tmdb_BE', 'tmdb_AT',
        'tmdb_TR', 'tmdb_PL', 'tmdb_CH', 'tmdb_XC', 'tmdb_FI', 'tmdb_NO',
        'tmdb_IR', 'tmdb_XG', 'tmdb_EG', 'tmdb_NG', 'tmdb_ZA'
    ]

    genre = [colonne for colonne in df_selection.columns if df_selection[colonne
    pays = [colonne for colonne in df_selection.columns if df_selection[colonne]

    index = df_ml.index
    df_ml_num = df_ml.select_dtypes('number')
    df_ml_cat = df_ml.select_dtypes(['object', 'category', 'string', 'bool'])

    # Normalisation des colonnes numériques
    SN = MinMaxScaler()
    df_ml_num_SN = pd.DataFrame(SN.fit_transform(df_ml_num), columns=df_ml_num.c

    df_ml_encoded = pd.concat([df_ml_num_SN, df_ml_cat], axis=1)

    # Création d'une liste de colonnes à utiliser pour le modèle
    caracteristiques = df_ml_encoded.columns.drop(['tconst', 'nconst', 'title',
        'Action', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime',
        'Documentary', 'Drama', 'Family', 'Fantasy', 'Game-Show', 'History',
        'Horror', 'Music', 'Musical', 'Mystery', 'News', 'Reality-TV',
        'Romance', 'Sci-Fi', 'Sport', 'Talk-Show', 'Thriller', 'War', 'Western',
        'tmdb_US', 'tmdb_FR', 'tmdb_GB', 'tmdb_DE', 'tmdb_JP', 'tmdb_IN',
        'tmdb_IT', 'tmdb_CA', 'tmdb_ES', 'tmdb_MX', 'tmdb_HK', 'tmdb_BR',
        'tmdb_SE', 'tmdb_SU', 'tmdb_PH', 'tmdb_KR', 'tmdb_AU', 'tmdb_CN',
        'tmdb_AR', 'tmdb_RU', 'tmdb_DK', 'tmdb_NL', 'tmdb_BE', 'tmdb_AT',
        'tmdb_TR', 'tmdb_PL', 'tmdb_CH', 'tmdb_XC', 'tmdb_FI', 'tmdb_NO',
        'tmdb_IR', 'tmdb_XG', 'tmdb_EG', 'tmdb_NG', 'tmdb_ZA'])

    # Sélection des films en fonction de la note
    bons_films = df_ml_encoded[df_ml_encoded['notes'] >= 0.7]

    # On veut que nos recommandations aient automatiquement un genre en commun e
```

```
    bons_films = bons_films[bons_films[genre].any(axis=1)] if genre else bons_fi
    bons_films = bons_films[bons_films[pays].any(axis=1)] if pays else bons_film

    # Création de notre modèle
    model = NearestNeighbors(n_neighbors=6, metric='euclidean')
    model.fit(bons_films[caracteristiques])

    # On déclare les caractéristiques du film sélectionné par l'utilisateur
    caract_film = df_ml_encoded[df_ml_encoded['tconst'] == tconst][caracteristiq

    # Calcul des distances et indices des voisins
    distances, indices = model.kneighbors(caract_film)

    # Affichage de la sélection des films en fonction des indices trouvés par le
    if caract_film['notes'].values[0] > 0.7:
        distances = distances[0][1:6]
        indices = indices[0][1:6]
        selection = bons_films.iloc[indices]['tconst']
    else:
        distances = distances[0][0:5]
        indices = indices[0][0:5]
        selection = bons_films.iloc[indices]['tconst']

    selection = pd.DataFrame(selection).reset_index(drop=True)

    # 2e reco : 5 films avec genre commun et pays différent

    # Sélection des films en fonction de la note
    bons_films2 = df_ml_encoded[df_ml_encoded['notes'] >= 0.7]

    # On veut que nos recommandations aient automatiquement un genre en commun e
    bons_films2 = bons_films2[bons_films2[genre].any(axis=1)] if genre else bons
    bons_films2 = bons_films2[~bons_films2[pays].any(axis=1)] if pays else bons_

    # Création de notre modèle
    model2 = NearestNeighbors(n_neighbors=6, metric='euclidean')
    model2.fit(bons_films2[caracteristiques])

    distances2, indices2 = model2.kneighbors(caract_film)

    # Affichage de la sélection des films en fonction des indices trouvés par le
    if caract_film['notes'].values[0] > 0.7:
        distances2 = distances2[0][1:6]
        indices2 = indices2[0][1:6]
        selection2 = bons_films2.iloc[indices2]['tconst']
    else:
        distances2 = distances2[0][0:5]
        indices2 = indices2[0][0:5]
        selection2 = bons_films2.iloc[indices2]['tconst']

    selection2 = pd.DataFrame(selection2).reset_index(drop=True)

    if selection.equals(selection2):
        return st.session_state["nb_selection"] = 1
    return afficher_resultats_similarite(selection, selection2)
```