Feuille TD n°2

Exercice en Machine Learning : Prédiction de la Note moyenne des livres

L'objectif de ce TD est de construire un modèle de machine learning pour prédire la note moyenne (average_rating) d'un livre en utilisant Elasticsearch. Nous explorerons les fonctionnalités de recherche et de comparaison de similarités pour créer des modèles de recommandation.

Exercices

1. Configuration d'Elasticsearch :

Connectez-vous à votre instance Elasticsearch en utilisant la bibliothèque elasticsearch.

```
# @title **Connexion à Elasticsearch**

es = Elasticsearch(
    hosts=['http://formakuntza-kibana.hupi.io:9200/'],
    http_auth=('elastic', 'BaudneOsGo41+')
)
# Tester la connexion :
try:
    # Récupérer les informations du cluster
    info = es.info()

    # Afficher des informations intéressantes
    print(f"Connexion réussie au cluster : {info['cluster_name']}")
    print(f"Version d'Elasticsearch : {info['version']['number']}")

    # Afficher les indices existants
    indices = es.cat.indices(format="json")
    print(f"Nombre d'index dans le cluster : {len(indices)}")

except Exception as e:
    print(f"Erreur de connexion : {e}")
```

2. Chargement du Dataset :

Utilisez la fonction Search pour lire l’ensemble des données de l’index “books”.

```
# @title **Lecture de l'index "books" comme les données d'entrées**

# @title **Afficher l'index "ecommerce"**
index_name = "ecommerce"

s = Search(using=es, index=index_name).query("match_all")

# Utilisation de la méthode scan pour récupérer tous les résultats rapidement
results = s.scan()

# Convertir les résultats en DataFrame
data = [hit.to_dict() for hit in results]
df = pd.json_normalize(data)

# Explorer les données
print(f"Show dataframe shape: {df.shape}")
print(f"Show dataframe first 10 rows: \n {df.head()}")
```

3. Exploration des Données :

Explorez les données pour comprendre les différentes caractéristiques présentes. Identifiez les colonnes pertinentes pour la prédiction de la note moyenne.

Le dataset provient d'un jeu de données opendata sur Kaggle, dont voici le lien : https://www.kaggle.com/datasets/abdallahwagih/books-dataset

Description des colonnes :

- **thumbnail :** URL de l'image miniature de la couverture du livre.
- **published_year :** Année de publication du livre.
- **num_pages :** Nombre total de pages dans le livre.
- **description :** Résumé ou description du livre.
- **average_rating :** Note moyenne du livre, basée sur les évaluations des utilisateurs.
- **title :** Titre du livre.
- **isbn13 :** ISBN-13, un identifiant unique à 13 chiffres pour les livres.
- **isbn10 :** ISBN-10, un identifiant unique à 10 chiffres pour les livres (ancien format).
- **categories :** Genre ou catégorie du livre.
- **ratings_count :** Nombre total d'évaluations données par les utilisateurs.
- **authors :** Auteur(s) du livre.
- **subtitle :** Sous-titre du livre, s'il en a un.

4. Créer une base de données d'entraînement et de test :

Vous devez créer un ensemble d'entraînement et de test. Vous enregistrerez l'ensemble d'entraînement dans un nouvel index qui s'appelera : uppa_2025_train_nom_prenom

5. Construction du Modèle de Similarité :

Utiliser la fonction *multi_match* d'Elasticsearch pour construire un modèle de similarité entre les livres. Comparer les résultats avec la fonction *more_like_this* pour évaluer les performances des deux approches.

Créer différents indicateurs afin de déterminer et prédire la note moyenne des livres en fonction des documents similaires trouvés.

```
##########################    Apply the multi_match function

response = es.search(index="books", body={
   "query": {
       "multi_match": {
           "query": "text to be evaluated",
           "fields": ["description"]
       }
   }
})

print(response)
hits = response['hits']['hits']
source_list = [hit['_source'] for hit in hits]
df = pd.json_normalize(source_list)
print(df)
```

```
#####################  Apply the more_like_this function

response = es.search(index="books", body={
   "query": {
       "more_like_this": {
           "fields": ["title", "author", "description"],
           "like": "text to be evaluated"
       }
   },
   "explain": "true"
})
```

```
print(response)
hits = response['hits']['hits']
source_list = [hit['_source'] for hit in hits]
df = pd.json_normalize(source_list)
print(df)
```

6. Construction du Modèle de Prédiction :

Divisez le dataset en ensembles d'entraînement et de test. Utilisez les notes moyennes comme variable cible et les résultats du modèle de similarité comme variables explicatives pour construire un modèle de détermination de la note.

Comparer les différents indicateurs (exemples : la note du document le plus similaire, une combinaison des notes des documents les plus similaires, …) créés afin de sélectionner le meilleur modèle.

7. Comparaison de Modèles :

Comparez les performances des modèles en termes de métriques de régression (MAE, MSE, RMSE, etc.).

8. Optimisation du Modèle :

Utilisez différentes variables explicatives pour affiner la recherche de similarité (par exemple, la langue du livre, catégorie des livres, …). Étudier et explorer les hyperparamètres et les fonctionnalités d'Elasticsearch pour optimiser les requêtes : “booster”, “filtrer”, … les documents lors de la comparaison de similarité.

Utilisez une fonction d'identification de la langue pour créer cette variable explicative, au choix :

- à partir de librairie Python
- à partir d'appelle du modèle de détection de langue d'Elasticsearch

Pistes

Utilisez la fonction more_like_this et multi_match pour construire les modèles de similarité.
Utilisez la bibliothèque langdetect pour identifier la langue des titres des livres.
