Feuille TD n°2

TP de Machine Learning : Prédiction de la Note moyenne des livres

L'objectif de ce TP 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. Chargement du Dataset :

Utilisez la bibliothèque pandas pour charger le dataset "Books Dataset" dans un DataFrame.

```
from elasticsearch import Elasticsearch
from google.colab import drive
drive.mount('/content/drive')
ls /content/drive/MyDrive/
```

2. Configuration d'Elasticsearch :

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

```
es = Elasticsearch(

hosts=["https://hupi-formation.es.europe-west9.gcp.elastic-cloud.com"],
   http_auth=('elastic', 'jRYMuKfRoMmCcC0NykeUtdTJ')
)
# Tester la connexion :
print(es.info())
```

3. Exploration des Données :

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

```
path_to_csv = '/content/drive/MyDrive/Books_Dataset.csv'
df = pd.read_csv(path_to_csv)
# Explorer les données
df.head()
```

4. 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)
```

5. 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.

6. Comparaison de Modèles :

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

7. Optimisation du Modèle :

Utilisez différentes variables explicatives pour affiner la recherche de similarité (par exemple, la langue du livre, catégorie des livres, …). Etudier 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.
