Projet de fin de module

Arabic Automated Short Answers Grading System for Moroccan History

Département Génie informatique

Module : Machine Learning

LSI2 2022-2023

Préparé par :

Mr. EL AACHAK Lotfi

Réalisé par :

MAZOZI Safae & Oualhaj Ouidad

Table des matières

I. Remerciement

Au terme de cette étude, nous tenons à exprimer nos profondes reconnaissances à notre professeur **M. EL AACHAK LOTFI** pour la confiance, la rigueur et les conseils encourageants qu'il nous a transmis tout au long du module de Machine Learning. Nous le remercions pour la richesse de ses remarques, ses conseils pertinents et sa disponibilité à servir. Son expérience partagée avec nous avec toute générosité et modestie a été très précieuse pour notre apprentissage.

Nous tenons également à souligner que le module de l'apprentissage automatique était un très bon cours. Les enseignements étaient clairs, pertinents et nous ont permis d'acquérir de nouvelles compétences que nous avons pu mettre en pratique dans ce projet.

II. Description de projet :

Arabic Automated short answers grading system for Moroccan history, the idea behind this system is to give the adequate grade to the students according to their answers, the system should be in Arabic, and you should prepare your own dataset.

• Scraping data from several sources: Arabic websites, Datasets, books, etc.

• Establishment of Arabic Natural language processing pipeline.

• Word embedding and encoding.

• Model Training based on classical machine learning algorithms.

• Evaluation of the models then the choice of the best one

• Model deployment and consumption via spa web application.

Specific Tools: NLTK, Word2vec, Glove, etc.

Tools: Sklearn, flask/fastapi, angular/react, scrapy, Doker, Github.

III. Introduction :

1) Clarification :

Ce projet intitulé "**Arabic Automated Short Answers Grading System for Moroccan History**" est un système très intéressant et nécessite l'utilisation de nombreuses techniques et outils pour sa réalisation.

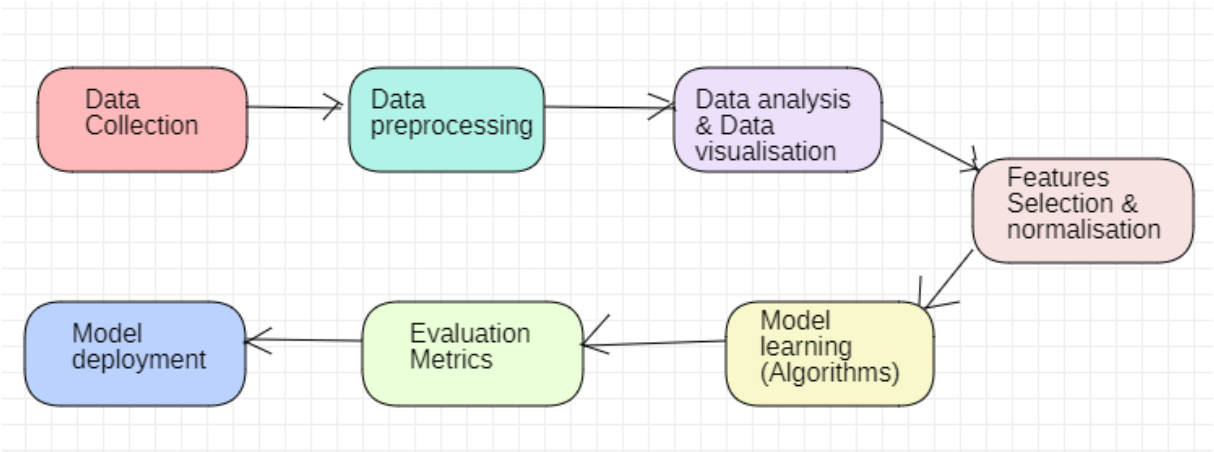
Il peut être divisé en deux grandes parties : une première partie consacrée à la **création de modèle de l'apprentissage automatique**, et une seconde partie dédiée au **déploiement de modèle** en utilisant une application complète (Backend et Frontend) avec une base de données.

- La partie d'apprentissage automatique est cruciale pour la création d'un modèle de machine learning capable de noter automatiquement les réponses courtes en arabe pour l'histoire marocaine.
- La deuxième partie est tout aussi importante, car elle permettra de déployer le modèle pour une utilisation pratique via une application web.

Nous avons donc travaillé sur ces deux parties pour réaliser ce projet ambitieux. Le résultat est un système de notation automatique des réponses courtes en arabe pour l'histoire

marocaine, qui peut être utilisé par les enseignants pour évaluer les réponses de leurs élèves de manière rapide et précise.

2) Schéma de projet :



3) Outils utilisés :

❖ Création de modèle de ML :

NLTK : signifie Natural Language Toolkit, qui est une plateforme populaire open-source pour construire des programmes Python pour travailler avec des données de langage humain. Il fournit une large gamme d'outils et de ressources pour diverses tâches de traitement de langage naturel, telles que la tokenisation, l'étiquetage des parties du discours, la reconnaissance des entités nommées, l'analyse de sentiment et l'apprentissage automatique pour la modélisation et la classification du langage.

Scikit-learn : est une bibliothèque open-source de machine learning pour Python. Elle offre un ensemble d'outils simples et efficaces pour la classification, la régression, le Clustering et la sélection de modèles en machine Learning, ….

Scikit-learn est utilisée dans ce projet pour le traitement de données, la normalisation, la division, l'évaluation, la régression, la similarité cosinus, …

Word2Vec : Word2Vec est une technique d'apprentissage non supervisé pour la représentation de mots en machine learning. Elle permet de représenter chaque mot d'un corpus de texte sous forme de vecteur numérique dense. Ces vecteurs sont calculés en utilisant des algorithmes de réseaux de neurones profonds.

Tandis que **Gensim** est une bibliothèque Python qui implémente cette méthode, ainsi que d'autres algorithmes de modélisation de texte.

Pandas : est une bibliothèque open-source pour la manipulation et l'analyse de données en Python. Elle fournit des structures de données hautement performantes pour stocker et manipuler des données tabulaires et des séries temporelles.

Numpy : est une bibliothèque open-source pour le calcul numérique en Python. Elle fournit un objet de tableau multidimensionnel de haute performance (ndarray), ainsi que des fonctions pour travailler avec ces tableaux.

❖ Déploiement de modèle

Flask : Flask est un framework web open-source pour Python qui permet de créer des applications web rapidement et facilement. Il est léger et extensible, et est souvent utilisé pour créer des API web et des applications web simples.

Angular : est un framework open-source développé par Google pour la création d'applications web dynamiques, basées sur le langage de programmation TypeScript. Il permet de créer des applications web de manière modulaire et structurée en utilisant des composants réutilisables.

MySQL : C'est un système de gestion de bases de données relationnelles. Il est largement utilisé dans les applications web pour stocker, organiser et manipuler des données structurées.

Partie 1 : Modèle d'apprentissage automatique

IV. Création de modèle :

1) Collection de données

-D'abord On a Collecté les données concernant l'historique marocaine en arabe à partir de différents sites web

-Après la recherche de données, on a les stocker dans des fichier texte pour la réviser et nettoyer.

-Après le nettoyage de données on a créé deux Datasets sous forme de fichiers **csv**, une pour stocker les questions et l'autre pour stocker les réponses et les scores.

-**dataset1** contient les colonnes : **Question_id** et **Question**

-**dataset2** contient les colonnes : **Question_id**, **Answer** et **Score**

-les deux Datasets ont une colonne commune est **Question_id**, pour identifier les questions.

-On a choisi 10 questions claires, après on a créé pour chaque question 100 réponses possibles (vrai, faux, moyen, …)

-On a associé à chaque réponse un score qui se varie de **0** à **5,** selon la performance de la réponse.

✓ Dataset 1 :

DataSet1.csv

F: > bureau > LSI > LSI-S4 > Machine-learning > projet > Data > DataSet1.csv

```
Question_id,Question_text
1,ما هي العلاقة بين المغرب و الدول الأوروبية خلال القرن التاسع عشر
2,ما هي العلاقة بين المغرب و الجزائر خلال حرب الصحراء الغربية
3,ما هي علاقة المغرب بالاتحاد الأفريقي
4,ما هي الحركة الوطنية في المغرب
5,ما هي الحضارات التي سكنت المغرب
6,ما هو نظام الحكم في المغرب
7,ما هي حركة الاستقلال في المغرب
8,من هم البربر
9,متى فرضت الحماية الفرنسيةعلى المغرب
10,ما هو معنى اسم المغرب
```

✓ Dataset 2 :

Dataset_2.csv

F: > bureau > LSI > LSI-S4 > Machine-learning > projet > Data > Dataset_2.csv

```
Question_id,Score,Answers
1,5,كانت العلاقة بين المغرب و الدول الأوروبية تتميز بالتوتر و الصراع حول الهيمنة على المغرب
1,5,كانت العلاقة بين المغرب و الدول الأوروبية تتميز بالتوتر و الصراع حول استعمار المغرب
1,5,كانت العلاقة بين المغرب و الدول الأوروبية تتميز بالتوتر و الصراع حول استيلاء على المغرب
1,5,كانت العلاقة بين المغرب و الدول الأوروبية تتميز بالتوتر و الصراع حول السيطرة على المغرب
1,5,العلاقة بين المغرب والدول الأوروبية كانت مشحونة بالتوتر والصراع حول الهيمنة على المغرب
1,5,كانت العلاقة بين المغرب والدول الأوروبية متوترة ومتصارعة حول الهيمنة على المغرب
1,5,تميزت العلاقة بين المغرب والدول الأوروبية بالتوتر والصراع حول السيطرة على المغرب
1,5,كانت العلاقة بين المغرب والدول الأوروبية ملتهبة بالتوتر والصراع حول السيطرة على المغرب
```

….

2) Chargement de données

✓ Charger les données au niveau de Jupyter à l'aide de la bibliothèque pandas de python :

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

```
# load data
df1 = pd.read_csv("DataSet1.csv")
df2 = pd.read_csv("DataSet2.csv")
```

✓ Fusionner les deux Datasets en une seule pour simplifier l'utilisation, à l'aide de l'id commune : Question_id

```
# Fusionner les deux ensembles de données en utilisant la colonne commune :
df = pd.merge(df1, df2, on='Question_id')
df.head()
```

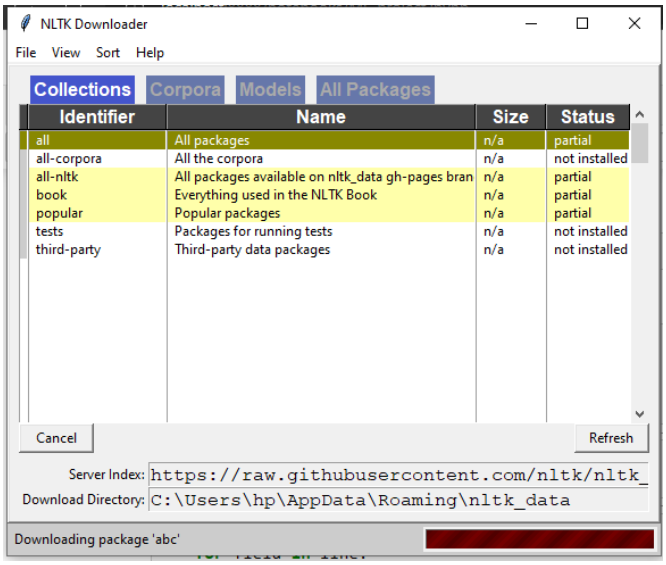
	Question_id	Question_text	Score	Answers
0	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	استعمار الدول الأوروبية المغرب
1	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	احتلال الدول الأوروبية للمغرب
2	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	الهيمنة الأوروبية على المغرب
3	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	3.0	الاستعباد الأوروبي للمغرب
4	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	الاحتلال الأوروبي للمغرب

3) Prétraitement des données

Les données collectées et chargées sur Jupyter doivent être nettoyées, normalisées et préparées pour l'analyse.

Cette étape est faite à l'aide de la bibliothèque NLTK de NLP (natural language processing)

✓ Installer NLTK



Après l'installation, il suffit d'importer le package avec l'instruction :

```
[4]: import nltk
```

Les étapes de prétraitement inclus les techniques suivantes : La tokenisation, la suppression des mots vides (les mots d'arrêt), le stemming et la lemmatisation

✓ Tokenisation :

La tokenisation est une technique de traitement de langage naturel qui consiste à diviser un texte en une séquence de "tokens" ou unités lexicales significatives, telles que les mots, les ponctuations et les chiffres. Le but de la tokenisation est de préparer un texte pour une analyse ultérieure.

On a établi la tokenisation seulement sur la colonne des réponses (pas les questions), car les réponses seront utilisées dans le traitement de modèle.

```
#Tokenisation
from nltk.tokenize import word_tokenize
df['Answers'] = df['Answers'].apply(nltk.word_tokenize)
```

Le texte après la tokenisation :

```
df['Answers'].head()
```

```
0    [كانت, العلاقة, بين, المغرب, و, الدول, الأورو...
1    [كانت, العلاقة, بين, المغرب, و, الدول, الأورو...
2    [كانت, العلاقة, بين, المغرب, و, الدول, الأورو...
3    [كانت, العلاقة, بين, المغرب, و, الدول, الأورو...
4    [العلاقة, بين, المغرب, والدول, الأوروبية, كانت...
Name: Answers, dtype: object
```

✓ Suppression des mots vides :

"mots vides" sont des mots très fréquents dans une langue donnée, mais qui n'apportent pas beaucoup d'informations pour l'analyse de texte. Ces mots sont supprimés lors du traitement de langage naturel, car ils ne contribuent pas significativement au sens du texte et peuvent entraîner des bruits indésirables dans les résultats d'analyse.

Les mots vides dans la langue Arabe :

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('Arabic'))
stop_words
```

```
, 'ء'}
, 'ءَ'
, 'آ'
, 'آب'
, 'آذار'
, 'آض'
, 'آمينَ'
, 'آناء'
, 'آنفا'
, 'آه'
, 'آها'
, 'آهاً'
```

Supprimer les mots vides de notre colonne des réponses :

```
#Drop stop words
from nltk.corpus import stopwords

stop_words = set(stopwords.words('Arabic'))
df['Answers'] = df['Answers'].apply(lambda x: [w for w in x if not w in stop_words])
```

✓ Stemming :

Le stemming est un processus de normalisation de mots dans le traitement automatique de langage naturel. Il s'agit de réduire les mots à leur racine ou à leur forme de base commune, en supprimant les affixes tels que les préfixes et les suffixes.

Form	Stem
saw	see (if token is verb)
saw	saw (if token is noun)

Raciner les mots dans la colonne des réponses :

```python
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
# Stem words in answers column
df['Answers'] = df['Answers'].apply(lambda x: [stemmer.stem(word) for word in x])
```

✓ Lemmatisation :

La lemmatisation est un processus de normalisation de mots, similaire au stemming, mais qui vise à réduire les mots à leur forme de base commune, en utilisant des règles grammaticales et des informations sur la morphologie des mots, plutôt que de simplement supprimer les affixes

```python
from nltk.stem.wordnet import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
# Lemmatize words in answers column
df['Answers'] = df['Answers'].apply(lambda x: [lemmatizer.lemmatize(word) for word in x])
```

La Dataset après le prétraitement :

	Question_id	Question_text	Score	Answers
0	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	[استعمار, الدول, الأوروبية, المغرب]
1	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	[احتلال, الدول, الأوروبية, للمغرب]
2	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	[الهيمنة, الأوروبية, المغرب]
3	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	3.0	[الاستعباد, الأوروبي, للمغرب]
4	1	ما هي العلاقة بين المغرب و الدول الأوروبية خلا...	4.0	[الاحتلال, الأوروبي, للمغرب]

4) Encodage des mots

Les données doivent être encodées en vecteurs de nombres afin de pouvoir être traitées par les algorithmes d'apprentissage automatique. Différentes techniques d'encodage, telles que l'encodage par sac de mots BOW, l'encodage TF-IDF et l'encodage **Word2Vec**, peuvent être utilisées.

On a choisi **Word2Vec** car cette technique d'encodage de texte est puissante et largement utilisée, qui est préférée pour sa capacité à capturer la sémantique des mots et les relations entre les mots dans un corpus de texte, ainsi que pour sa performance élevée dans de nombreuses tâches de traitement de texte.

Contrairement à l'encodage par BOW et à l'encodage TF-IDF, qui ne prennent en compte que la fréquence d'apparition des mots dans un texte

✓ Créer le modèle

```
#Word2Vec
from gensim.models import Word2Vec
model = Word2Vec(df['Answers'], min_count=1)
```

from gensim.models import Word2Vec : importe la classe Word2Vec à partir de la bibliothèque Gensim.

model = Word2Vec(df['Answers'], min_count=1) : crée un nouvel objet de modèle Word2Vec en utilisant les réponses stockées dans la colonne "Answers" du DataFrame . Le paramètre min_count=1 signifie que les mots qui apparaissent au moins une fois dans le corpus de texte seront inclus dans le modèle.

✓ Vectoriser les réponses :

```
#Vectorise toutes les réponses
max_len = model.vector_size

W = np.zeros((len(df['Answers']), max_len))
for i, sentence in enumerate(df['Answers']):
    word_vectors = []
    for word in sentence:
        if word in model.wv:
            word_vectors.append(model.wv[word])
    if len(word_vectors) > 0:
        mean_vector = np.mean(word_vectors, axis=0)
        W[i, :] = mean_vector
```

➔ Ce code vectorise toutes les réponses stockées dans la colonne "Answers" du Dataframe en utilisant le modèle Word2Vec entraîné précédemment, puis il stocker les vecteurs dans le tableau numpy W

✓ Vocabulaire de mots :

```
# get a list of all words in the vocabulary
vocab_list = model.wv.index_to_key

# get a dictionary mapping each word to its index in the word vectors
vocab_dict = model.wv.key_to_index
print(vocab_dict)
```

```
'البربر': 5, 'نظام': 6, 'العلاقة': 7, 'إفريقيا': 8, 'الفرنسية': 9, 'الأوروبية': 10, 'سياسي
'الصحراء': 16, 'تم': 17, 'الأفريقي': 18, 'كانت': 19, 'تشريعيين': 20, 'الاتحاد': 21,
26, 'ديمقراطي': 27, 'الغرب': 28, 'دستوري': 29, 'للمغرب': 30, 'الحركة': 31, 'توقيع
'الحصول': 37, 'العربية': 38, 'كلمة': 39, 'حول': 40, 'مجلس': 41, 'الحكم': 42, 'ال
'العرب': 48, 'تعني': 49, '30': 50, 'باللغة': 51, 'الوطنية': 52, 'حضارات': 53,
'المستشارين': 59, 'الحفيظ': 60, 'تسعى': 61, 'عبد': 62, 'النواب': 63, 'السلطان':
'الأمازيغ': 69, 'علاقة': 71, 'السكان': 72, 'التاسع': 73, '2017': 74, 'عائد
'السيطرة': 80, 'الهيمنة': 81, 'البلاد': 82, 'باسم': 83, 'الحضارات': 84, 'سكن': 85,
'الانسحاب': 90, 'يطالب': 91, 'والصراع': 92, 'لشمال': 94, 'يتميز': 95, 'مسد
```

➔ Ce code extrait la liste de tous les mots du vocabulaire et un dictionnaire qui mappe chaque mot à son index dans les vecteurs de mots du modèle Word2Vec entraîné précédemment.

✓ Tester le modèle Word2Vec :

On peut tester la performance de modèle manuellement :

```
#Tester word2vec
word = model.wv['الاستعمار']
print(word)
```

```
[-0.03194354  0.0357979   0.02319086 -0.0173836  -0.03002614 -0.12046469
  0.04121247  0.11332423 -0.07845246 -0.03308956 -0.06177983 -0.07139444
 -0.04568893  0.01330139  0.04039284 -0.03884895  0.00198371 -0.06906544
 -0.01391425 -0.14153911  0.02472244  0.0699456   0.00604487 -0.01572143
 -0.00721072  0.03270558 -0.02058942 -0.05325682 -0.07702371  0.00627368
  0.07168656 -0.01338417  0.02527935 -0.06058587 -0.05235137  0.07489517
  0.0296154  -0.05678855 -0.03357296 -0.13980284 -0.02681108 -0.03551292
 -0.08383255  0.01101691  0.0716973  -0.01206623 -0.07077386  0.00631023
```

```
#Test word2vec
# calculate the similarity between two words
word1 = "الفرنسية"
word2 = "الفرنسية"

similarity = model.wv.similarity(word1, word2)
print(f"Similarity between {word1} and {word2} is: {similarity}")
```

```
Similarity between الفرنسية and الفرنسية is: 1.0
```

5) Entraînement du modèle

Les données prétraitées et encodées peuvent être utilisées pour entraîner différents modèles d'apprentissage automatique, tels que les modèles de régression linéaire, les modèles de classification ou les modèles de réseau de neurones.

a. Choisir l'algorithme :

- On a choisi de travailler avec les modèles de **régression**, ils sont plus performants que les modèles de classification dans notre cas.
- On a testé plusieurs algorithmes de régression comme : **Linear Regression**, **GradientBoostingRegressor**, **MLPRegressor**, **RandomForestRegressor**, et on a constaté que tous ces modèles donnent des résultats (scores) logiques et très proches au scores actuels, mais les plus précis sont **RandomForestRegressor** et **GradientBoostingRegressor**
- **RandomForestRegression** (ou Régression avec les Forêts Aléatoires) est un algorithme d'apprentissage supervisé utilisé pour résoudre des problèmes de régression. Il fait partie de la famille des algorithmes d'ensemble, qui combinent les prédictions de plusieurs modèles individuels pour améliorer les performances de prédiction.
- **GradientBoostingRegression** est un algorithme d'apprentissage supervisé qui combine plusieurs arbres de décision en cascade pour résoudre des problèmes de régression. Il utilise la descente de gradient pour trouver les poids optimaux pour chaque arbre et minimiser la fonction de coût globale.

b. Techniques supplémentaires :

On a aussi essayé de combiner l'algorithme de régression avec la mesure de similarité cosinus pour tester s'il a une amélioration des performances de prédiction.

- **La similarité cosinus** est une mesure de similarité entre deux vecteurs. Elle est souvent utilisée pour calculer la similarité entre des documents ou des textes dans le domaine du traitement de texte. La formule pour calculer la similarité cosinus entre deux vecteurs A et B est la suivante :

  Similarity = (A. B) / (||A|| * ||B||)

- On a utilisé cette technique pour calculer la similarité cosinus entre la réponse de l'élève et les réponses existantes de la même question donnée.

c. Entrainer les modèles choisis :

Ici on a donné une réponse avec question_id pour tester les différents modèles :

```
# Demander à l'utilisateur d'entrer une réponse d'une question donnée
question_id= 1
response = "استعمار الدول الأوروبية المغرب"
```

➔ **Cette réponse est très similaire à une réponse dans df à un score de 4/5.**

1. Prétraiter et vectoriser la réponse donnée de la même manière que la colonne Answers :

```
# Tokeniser la réponse entrée
response_tokens = nltk.word_tokenize(response)

# Supprimer les mots d'arrêt de la réponse
response_tokens = [w for w in response_tokens if not w in stop_words]

# Appliquer le stemming à la réponse
response_tokens = [stemmer.stem(word) for word in response_tokens]

# Appliquer la lemmatisation à la réponse
response_tokens = [lemmatizer.lemmatize(word) for word in response_tokens]

# Vectoriser la réponse en utilisant le modèle Word2Vec
response_vector = np.mean([model.wv[word] for word in response_tokens if word in model.wv], axis=0)

response_tokens
```

```
['استعمار', 'الدول', 'الأوروبية', 'المغرب']
```

2. Chercher les réponses existantes dans df de la question ayant l'id donnée :

```
def find_existing_answers_vectors(df, question_id, W):

    if question_id in df['Question_id'].values:
        # Trouver tous les indices correspondants à l'ID de la question
        question_indices = df.index[df['Question_id'] == question_id].tolist()

        # Extraire les vecteurs de réponse correspondants à partir de la matrice W
        existing_answers_vectors = []
        for i in question_indices:
            existing_answers_vectors.append(W[i])
    else:
        print("L'ID de la question que vous avez fourni est incorrect")
        existing_answers_vectors = None
    existing_answers_vectors = np.array(existing_answers_vectors)
    return existing_answers_vectors
```

❖ Prédire le score de la réponse donnée en utilisant la **similarité cosinus** :

```
# cosin similarity
from sklearn.metrics.pairwise import cosine_similarity

# Trouver les réponses existantes correspondant à la question donnée
existing_answers_vectors = find_existing_answers_vectors(df, question_id, W)

# Calculer la similarité cosinus entre la réponse donnée et les réponses existantes
response_vector_reshaped = response_vector.reshape(1, -1)
cosine_similarities = cosine_similarity(response_vector_reshaped, existing_answers_vectors)

# Obtenir les scores correspondant aux réponses existantes
existing_scores = df[(df['Question_id'] == question_id)]['Score']

# Trouver l'index de la réponse la plus similaire
most_similar_index = np.argmax(cosine_similarities)

# Retourner le score correspondant à la réponse la plus similaire
predicted_score = existing_scores.iloc[most_similar_index]
predicted_score
```

```
4.0
```

➔ **Remarquons que la similarité cosinus est très efficace mais ce n'est pas un algorithme d'apprentissage automatique !**

❖ Prédire le score de la réponse donnée en utilisant **RandomForestRegressor** et **la similarité cosinus** :

```
# RandomForestRegressor + cosin similarity
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import numpy as np

existing_answers_vectors = find_existing_answers_vectors(df, question_id, W)

# Initialize an empty array to store the cosine similarities
cosine_similarities = []

# Loop over the existing answer vectors and compute cosine similarity with the response vector
for answer_vector in existing_answers_vectors:
    cosine_similarity_score = cosine_similarity(response_vector.reshape(1, -1), answer_vector.reshape(1, -1))
    cosine_similarities.append(cosine_similarity_score[0][0])

# Convert the cosine similarities to a NumPy array
cosine_similarities = np.array(cosine_similarities)
```

```
# Concatenate the cosine similarities with the existing answer vectors
combined_features = np.column_stack((existing_answers_vectors, cosine_similarities))

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(combined_features, df[df['Question_id'] == question_id]['Score']

# Scale input data to [0, 1] range
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train the Random Forest Regression model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)
```

```
# Reshape response vector to have the same shape as existing answer vectors
response_vector_reshaped = np.tile(response_vector, (existing_answers_vectors.shape[0], 1))

# Initialize an empty array to store the response cosine similarities
response_cosine_similarities = []

# Loop over the existing answer vectors and compute cosine similarity with the response vector
for answer_vector in existing_answers_vectors:
    response_cosine_similarity_score = cosine_similarity(response_vector_reshaped, answer_vector.reshape(1, -1))
    response_cosine_similarities.append(response_cosine_similarity_score[0][0])

# Convert the response cosine similarities to a NumPy array
response_cosine_similarities = np.array(response_cosine_similarities)

# Reshape response_cosine_similarities to match the shape of response_vector_reshaped
response_cosine_similarities = response_cosine_similarities.reshape(-1, 1)
```

```
# Concatenate the cosine similarities with the response vector
response_combined_features = np.column_stack((response_vector_reshaped, response_cosine_similarities))

# Scale the response vector
response_combined_features_scaled = scaler.transform(response_combined_features)

# Predict the score using the Random Forest Regression model
score_prediction = rf_model.predict(response_combined_features_scaled)

# Print the predicted score
print("Predicted score:", score_prediction[0])
```

```
Predicted score: 3.08
```

Tout d'abord, le code calcule la similarité cosinus entre le vecteur de réponse et les vecteurs de réponse existants. Les similarités cosinus sont ensuite concaténées avec les vecteurs de réponse existants. La matrice de caractéristiques résultante est divisée en ensembles d'entraînement et de test, mise à l'échelle en utilisant MinMaxScaler, et utilisée pour entraîner un modèle de régression de la forêt aléatoire.

Ensuite, le code calcule la similarité cosinus entre le vecteur de réponse et les vecteurs de réponse existants et la concatène avec le vecteur de réponse. Ce vecteur de caractéristiques combiné est mis à l'échelle en utilisant la même mise à l'échelle utilisée pour l'entraînement, puis utilisé pour prédire le score en utilisant le modèle de régression de la forêt aléatoire entraîné.

Enfin, le score prédit est imprimé. Dans l'ensemble, ce code effectue une tâche de régression basée sur la similarité, où la similarité entre le vecteur de réponse et les vecteurs de réponse existants est utilisée comme caractéristique pour prédire le score de la réponse.

- Prédire le score de la réponse donnée en utilisant **GradientBoostingRegressor** et **la similarité cosinus** :

```python
# GradientBoostingRegressor + cosine_similarity
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import numpy as np

existing_answers_vectors = find_existing_answers_vectors(df, question_id, W)

# Initialize an empty array to store the cosine similarities
cosine_similarities = []

# Loop over the existing answer vectors and compute cosine similarity with the response vector
for answer_vector in existing_answers_vectors:
    cosine_similarity_score = cosine_similarity(response_vector.reshape(1, -1), answer_vector.reshape(1, -1))
    cosine_similarities.append(cosine_similarity_score[0][0])

# Convert the cosine similarities to a NumPy array
cosine_similarities = np.array(cosine_similarities)

# Concatenate the cosine similarities with the existing answer vectors
combined_features = np.column_stack((existing_answers_vectors, cosine_similarities))
```

```python
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(combined_features, df[df['Question_id'] == question_id]['Score']

# Scale input data to [0, 1] range
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train the GradientBoostingRegressor model
gb_model = GradientBoostingRegressor(random_state=42)
gb_model.fit(X_train_scaled, y_train)

# Reshape response vector to have the same shape as existing answer vectors
response_vector_reshaped = np.tile(response_vector, (existing_answers_vectors.shape[0], 1))

# Initialize an empty array to store the response cosine similarities
response_cosine_similarities = []

# Loop over the existing answer vectors and compute cosine similarity with the response vector
for answer_vector in existing_answers_vectors:
    response_cosine_similarity_score = cosine_similarity(response_vector_reshaped, answer_vector.reshape(1, -1))
    response_cosine_similarities.append(response_cosine_similarity_score[0][0])
```

```python
# Convert the response cosine similarities to a NumPy array
response_cosine_similarities = np.array(response_cosine_similarities)

# Reshape response_cosine_similarities to match the shape of response_vector_reshaped
response_cosine_similarities = response_cosine_similarities.reshape(-1, 1)

# Concatenate the cosine similarities with the response vector
response_combined_features = np.column_stack((response_vector_reshaped, response_cosine_similarities))

# Scale the response vector
response_combined_features_scaled = scaler.transform(response_combined_features)

# Predict the score using the GradientBoostingRegressor model
score_prediction = gb_model.predict(response_combined_features_scaled)

# Print the predicted score
print("Predicted score:", score_prediction[0])
```

```
Predicted score: 3.9837534086793878
```

Tout d'abord, le code calcule la similarité cosinus entre le vecteur de réponse et les vecteurs de réponse existants. Les similarités cosinus sont ensuite concaténées avec les vecteurs de réponse existants. La matrice de caractéristiques résultante est divisée en ensembles d'entraînement et de test, mise à l'échelle en utilisant MinMaxScaler, et utilisée pour entraîner un modèle de GradientBoostingRegressor.

Ensuite, le code calcule la similarité cosinus entre le vecteur de réponse et les vecteurs de réponse existants et la concatène avec le vecteur de réponse. Ce vecteur de caractéristiques combiné est mis à l'échelle en utilisant la même mise à l'échelle utilisée pour l'entraînement, puis utilisé pour prédire le score en utilisant le modèle de GradientBoostingRegressor entraîné.

Enfin, le score prédit est imprimé. Dans l'ensemble, ce code effectue une tâche de régression basée sur la similarité, où la similarité entre le vecteur de réponse et les vecteurs de réponse existants est utilisée comme caractéristique pour prédire le score de la réponse.

- Prédire le score de la réponse donnée en utilisant **GradientBoostingRegressor** : (**le meilleur modèle dans notre cas**)

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import GradientBoostingRegressor

existing_answers_vectors = find_existing_answers_vectors(df, question_id, W)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(existing_answers_vectors, df[(df['Question_id'] == question_id)]['Score'],

# Create and train the Gradient Boosted Regression model
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)

# Reshape response vector to have shape (1, n)
response_vector_reshaped = response_vector.reshape(1, -1)

# Predict the score using the Gradient Boosted Regression model
score_prediction = gb_model.predict(response_vector_reshaped)

# Print the predicted score
print("Predicted score:", score_prediction[0])
```

```
Predicted score: 4.000138097762091
```

d. Remarque :

-Remarquons que le score prédit par l'algorithme **GradientBoostingRegressor (4.00) ou GradientBoostingRegressor + similarité cosinus (3.98)** est très précis et proche de score actuel de la réponse **(4.0).**

-De même pour l'algorithme **RandomForestRegressor** + **similarité cosinus (3.08**), mais il est moins proche que les autres.

-La similarité cosinus seule donne des bons résultats si la réponse sonnée et existe dans df, sinon elle ne donne pas des meilleures prédictions !

6) Choix du meilleur modèle

Nous pouvons observer que les modèles GradientBoostingRegressor et GradientBoostingRegressor + similarité cosinus donnent des scores prédits très précis, se rapprochant du score actuel de la réponse.

De même, l'utilisation de RandomForestRegressor en combinaison avec la similarité cosinus donne également un score prédit assez proche.

Il est important de noter que l'utilisation de la similarité cosinus seule donne de bons résultats uniquement si la réponse est présente dans df. Dans le cas contraire, les prédictions ne seront pas aussi précises.

En conséquence, **GradientBoostingRegressor** est le meilleur modèle pour obtenir des très bonnes prédictions.

✓ Une autre méthode de choisir le meilleur modèle et le score R2 pour évaluer la performance de chaque modèle :

```
# RandomForestRegressor R2 score
from sklearn.metrics import r2_score

y_test_pred = rf_model.predict(X_test_scaled)
r2 = r2_score(y_test, y_test_pred)
print("RandomForestRegressor R2 score:", r2)
```

```
RandomForestRegressor R2 score: 0.7187188776765937
```

```
# MLPRegressor R2 score
y_test_pred = mlp_model.predict(X_test_scaled)
r2 = r2_score(y_test, y_test_pred)

print("MLPRegressor R2 score :", r2)
```

```
MLPRegressor R2 score : 0.5182144306996652
```

```
# GradientBoostingRegressor r2 score
y_test_pred = gb_model.predict(X_test)
r2 = r2_score(y_test, y_test_pred)
print("GradientBoostingRegressor R2 score :", r2)
```

```
GradientBoostingRegressor R2 score : 0.7009468619722126
```

Partie 2 : Déploiement de modèle

V. Backend

1. Dépendances principales :

Vérifier l'installation de python :

```
C:\Users\hp>python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Après avoir installé Python, il faut installer l' **pipenv** (Cet outil vise à apporter le meilleur de tous les mondes de packaging (bundler, composer, npm, etc.) aux développeurs Python.)

```
C:\Users\hp>pip install pipenv
Requirement already satisfied: pipenv in c:\users\hp\appdata\local\programs\python\python310\lib\site-packages (2023.4.2
9)
Requirement already satisfied: virtualenv-clone>=0.2.5 in c:\users\hp\appdata\local\programs\python\python310\lib\site-p
ackages (from pipenv) (0.5.7)
```

➔ Python et pipenv ensemble suffisent pour commencer à développer notre application Flask.

Cependant, comme nous souhaitons conserver les données transactionnelles, nous devons toujours choisir et configurer un moteur de base de données.

Pour faciliter la vie, nous utiliserez **SQLAlchemy** pour persister et récupérer les données du moteur choisi.

Nous pourrons facilement nous connecter et utiliser n'importe quel moteur de base de données SQL majeur (par exemple **MySQL**).

Remarque :

SQLAlchemy : C'est une bibliothèque open-source en Python qui fournit une interface pour travailler avec des bases de données relationnelles. Elle permet de représenter des tables de base de données en tant qu'objets Python, de créer des requêtes SQL de manière programmatique, et de gérer les connexions à la base de données de manière transparente.

SQLAlchemy offre une grande flexibilité en termes de gestion des relations entre les tables, de gestion des transactions, et de gestion des résultats de requêtes. Elle prend en charge de nombreux types de bases de données, notamment PostgreSQL, MySQL, Oracle, SQLite, et Microsoft SQL Server.

Installer SQLAlchemy :

```
C:\Users\hp>pip install sqlalchemy
Collecting sqlalchemy
  Downloading SQLAlchemy-2.0.15-cp310-cp310-win_amd64.whl (2.0 MB)
     ---------------------------------------- 2.0/2.0 MB 47.0 kB/s eta 0:00:00
Collecting greenlet!=0.4.17
  Downloading greenlet-2.0.2-cp310-cp310-win_amd64.whl (192 kB)
     ---------------------------------------- 192.2/192.2 KB 80.3 kB/s eta 0:00:00
Collecting typing-extensions>=4.2.0
  Downloading typing_extensions-4.5.0-py3-none-any.whl (27 kB)
Installing collected packages: typing-extensions, greenlet, sqlalchemy
Successfully installed greenlet-2.0.2 sqlalchemy-2.0.15 typing-extensions-4.5.0
```

Pour utiliser SQLAlchemy avec une base de données particulière comme MySQL, il faut également installer le pilote approprié pour MySQL :

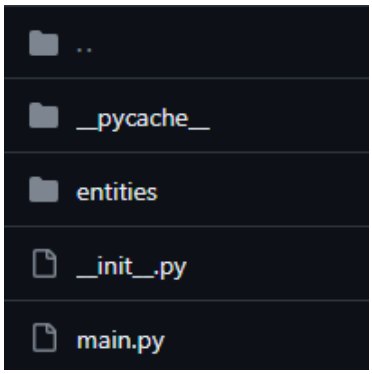
```
C:\Users\hp>pip install pymysql
Collecting pymysql
  Downloading PyMySQL-1.0.3-py3-none-any.whl (43 kB)
     ---------------------------------------- 43.7/43.7 KB ? eta 0:00:00
Installing collected packages: pymysql
Successfully installed pymysql-1.0.3
```

Une fois que **SQLAlchemy** et le pilote **pymysql** sont bien installés, il faut configurer l'application et commencer à utiliser SQLAlchemy pour interagir avec la base de données.

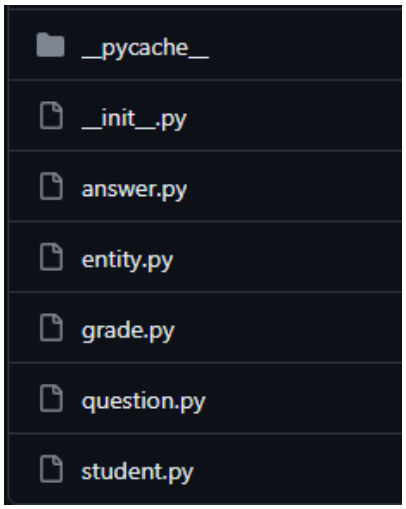
Mais d'abord on va définir la structure générale de l'application Flask :

2. Structure de l'application :

Voici la structure finale de notre application Flask de backend :



❖ **Les entités :**



Une fois que **SQLAlchemy** et le pilote **pymysql** sont bien installés, il faut configurer l'application et commencer à utiliser SQLAlchemy pour interagir avec la base de données.

```python
from sqlalchemy import create_engine, Column, String, Integer, DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

db_url = 'localhost:3306'
db_name = 'online-exam'
db_user = 'root'
db_password = ''
engine = create_engine(f'mysql://{db_user}:{db_password}@{db_url}/{db_name}')
Session = sessionmaker(bind=engine)

Base = declarative_base()
```

Maintenant il faut définir les modèles pour nos tables de base de données (les entités) en spécifiant les identifiants, les colonnes de chaque table et les relations entre les tables à l'aide de SQLAlchemy.

✓ Classe Entity :

```python
class Entity():
    id = Column(Integer, primary_key=True)
    created_at = Column(DateTime)
    updated_at = Column(DateTime)

    def __init__(self):
        self.created_at = datetime.now()
        self.updated_at = datetime.now()
```

La classe est un modèle de base qu'on a utilisé pour définir d'autres modèles pour les tables de notre base de données. Cette classe contient des colonnes qui sont couramment utilisées dans de nombreuses tables de base de données, telles que les colonnes id, created_at, updated_at, et last_updated_by.

✓ Classe Student :

```python
class Student(Entity, Base):
    __tablename__ = 'students'
    first_name = Column(String(255))
    last_name = Column(String(255))
    email = Column(String(255))

    answers = relationship("Answer", backref="students")
    grades = relationship("Grade", backref="students")

    def __init__(self, first_name,last_name,email):
        Entity.__init__(self)
        self.first_name = first_name
        self.last_name = last_name
```

```
        self.email = email

class StudentSchema(Schema):
    id = fields.Integer()
    first_name = fields.Str()
    last_name=fields.Str()
    email=fields.Str()
    created_at = fields.DateTime()
    updated_at = fields.DateTime()
```

✓ Classe Grade:

```
class Grade(Entity, Base):
    __tablename__ = 'grades'
    id = Column(Integer, primary_key=True)
    answer_id = Column(Integer, ForeignKey('answers.id'))
    student_id = Column(Integer, ForeignKey('students.id'))
    score = Column(Float)

    def __init__(self, score, answer_id, student_id):
        Entity.__init__(self)
        self.score = score
        self.answer_id = answer_id
        self.student_id= student_id

class GradeSchema(Schema):
    id = fields.Integer()
    student_id= fields.Integer()
    score=fields.Float()
    answer_id=fields.Integer()
    created_at = fields.DateTime()
    updated_at = fields.DateTime()
```

✓ Classe Question :

```
class Question(Entity, Base):
    __tablename__ = 'questions'
    text_question = Column(String(255))

    answers = relationship("Answer", backref="questions")

    def __init__(self, text_question):
        Entity.__init__(self)
        self.text_question = text_question
class QuestionSchema(Schema):
id = fields.Number()
text_question = fields.Str()
created_at = fields.DateTime()
```

```
updated_at = fields.DateTime()
```

✓ Classe Answer :

```
class Answer(Entity, Base):
    __tablename__ = 'answers'
    id = Column(Integer, primary_key=True)
    text_answer = Column(String(255))
    question_id = Column(Integer, ForeignKey('questions.id'))
    student_id = Column(Integer, ForeignKey('students.id'))
    grades = relationship("Grade", backref="answers")

    def __init__(self, text_answer, question_id ,student_id):
        Entity.__init__(self)
        self.text_answer = text_answer
        self.question_id = question_id
        self.student_id = student_id

class AnswerSchema(Schema):
    id = fields.Integer(dump_only=True)
    text_answer = fields.Str()
    question_id= fields.Integer()
    student_id=fields.Integer()
    created_at = fields.DateTime()
    updated_at = fields.DateTime()
```

Maintenant après la création de toutes les entités et la configuration de la base de données, et aussi de définir les Différents Schémas

- On a créé le fichier **./src/main.py** pour exposer les points d'accès basés sur SQLAlchemy et mySql . Ces points d'accès doivent être capables de recevoir des requêtes POST pour créer de nouvelles instances des classes, et de recevoir des requêtes GET pour sérialiser ces instances en tant qu'un tableau JSON.

```
# coding=utf-8
from flask import Flask, jsonify, request
from flask_cors import CORS
from .entities.entity import Session, engine, Base
from .entities.question import Question, QuestionSchema
from .entities.answer import Answer, AnswerSchema
from .entities.grade import Grade
from .entities.grade import GradeSchema
```

```python
# creating the Flask application
app = Flask(__name__)
# ... other import statements .
CORS(app)


# if needed, generate database schema
Base.metadata.create_all(engine)
```

Questions:

- Récupération des questions depuis la base de données

```python
#-----------------------------------------------questions-----------------------------------
@app.route('/questions')
def fetchQuestions():
    # fetching from the database
    session = Session()
    question_objects = session.query(Question).all()

    # transforming into JSON-serializable objects
    schema = QuestionSchema(many=True)
    questions = schema.dump(question_objects)

    # serializing as JSON
    session.close()
    return jsonify(questions)
```

Answers :

- Dans la fonction **submitAnswers** On a appelé la fonction **vectoriseRespose** Pour vectoriser la réponse de l'étudiant, après On a appelé la fonction **find_existing_answers** pour récupérer tous les réponses (qui sont déjà vectorisé) en spécifiant le **question_id** pour les donner au model comme données d'entrainement Et En fin on appelle la fonction **prédiction** qui prend comme arguments le retour de la fonction find_existing_answers (données d'entrainements) et la réponse vectorisé de l'étudiant , et le question_id pour récupérer les scores des réponses d'entrainements ,Et comme ça la fonction va prédire le score de réponse

```python
@app.route('/answers', methods=['POST'])
def submitAnswers():
    # mount answer object
    posted_answer = AnswerSchema(only=('text_answer', 'question_id', 'student_id')).load(request.get_json(), many=False)
    answer = Answer(**posted_answer)

    session = Session()
    session.add(answer)
    session.commit()
    #Calling  verctoriseResponse , find_existing_answers_vectors , prediction Functions
    response_vector = verctoriseResponse(answer.text_answer)
    existing_answers_vectors = find_existing_answers_vectors(df,answer.question_id,W)
    predicted_score = prediction(response_vector,existing_answers_vectors,answer.question_id)
    print("predicted score :", predicted_score)
    submitGrades(score=predicted_score, answer_id=answer.id, student_id=answer.student_id)
    print("grades submitted")
    # return created answer
    answer_schema = AnswerSchema()
    response = answer_schema.dump(answer)

    session.close()
    return jsonify(response), 201
```

Ma

- Récupération des réponses :

```python
@app.route('/answers')
def fetchAnswers():
    # fetching from the database
    session = Session()
    Answer_objects = session.query(Answer).all()

    # transforming into JSON-serializable objects
    schema = AnswerSchema(many=True)
    answers = schema.dump(Answer_objects)

    # serializing as JSON
    session.close()
    return jsonify(answers)
```

Grade :

- Après que le modèle prédit la réponse, il donne un score, ici on appelle la fonction submitGrade qui prend comme arguments le score et aussi answer_id et student_id pour stocker la note de réponse

```python
#--------------------------------------------------------grades----------------------------------------------
def submitGrades(score, answer_id, student_id):
    # create a dictionary with the attribute values
    posted_grade = {
        'score': score,
        'answer_id': answer_id,
        'student_id': student_id

    }
    print()
    # mount grade object
    grade = Grade(**posted_grade)

    session = Session()
    try:
        session.add(grade)
        session.commit()
    except Exception as e:
        session.rollback()
        print("Error while adding grade to database:", str(e))
        return jsonify({"error": "Failed to add grade to database"}), 500

    # return created grade
    grade_schema = GradeSchema()
    response = grade_schema.dump(grade)
    session.close()
    return jsonify(response), 201
```

- Récupération des notes

```python
@app.route('/grades')
def fetchGrades():
    # fetching from the database
    session = Session()
    Grade_objects = session.query(Grade).all()

    # transforming into JSON-serializable objects
    schema = GradeSchema(many=True)
    grades = schema.dump(Grade_objects)

    # serializing as JSON
    session.close()
    return jsonify(grades)
```

- On a copié collé le code Jupyter du modèle d'entrainement dans main.py, et on a installé les bibliothèques dont on a besoin

Machine le

```python
#----------------------------------------ML Code ------------------------------------
# apload data
df2 = pd.read_csv("C:/Users/safae/Documents/1sIS4/ML2/Dataset_Small_range.csv")
df1 = pd.read_csv("C:/Users/safae/Documents/1sIS4/ML2/new_DATASET1 (1).csv")

# Fusionner les deux ensembles de données en utilisant la colonne commune : Question_id
df = pd.merge(df1, df2, on='Question_id')
df.head()
#Tokenisation
df['Answers'] = df['Answers'].apply(nltk.word_tokenize)

#Drop stop words
stop_words = set(stopwords.words('Arabic'))
df['Answers'] = df['Answers'].apply(lambda x: [w for w in x if not w in stop_words])

#Lemmatization & stemming
stemmer = PorterStemmer()
# Stem words in answers column
df['Answers'] = df['Answers'].apply(lambda x: [stemmer.stem(word) for word in x])

lemmatizer = WordNetLemmatizer()
# Lemmatize words in answers column
df['Answers'] = df['Answers'].apply(lambda x: [lemmatizer.lemmatize(word) for word in x])

#df.head()
#Word2Vec
model = Word2Vec(df['Answers'], min_count=1)
#Vectorise toutes les réponses
max_len = model.vector_size

W = np.zeros((len(df['Answers']), max_len))
for i, sentence in enumerate(df['Answers']):
    word_vectors = []
    for word in sentence:
        if word in model.wv:
            word_vectors.append(model.wv[word])
    if len(word_vectors) > 0:
        mean_vector = np.mean(word_vectors, axis=0)
        W[i, :] = mean_vector
```

- Fonction de la Vectorisation de la réponse de l'étudiant :

```python
#Vectorise la réponse d'etudiant
def verctoriseResponse(response):
    # Tokeniser la réponse entrée
    response_tokens = nltk.word_tokenize(response)

    # Supprimer les mots d'arrêt de la réponse
    response_tokens = [w for w in response_tokens if not w in stop_words]

    # Appliquer le stemming à la réponse
    response_tokens = [stemmer.stem(word) for word in response_tokens]

    # Appliquer la lemmatisation à la réponse
    response_tokens = [lemmatizer.lemmatize(word) for word in response_tokens]

    # Vectoriser la réponse en utilisant le modèle Word2Vec
    response_vector = np.mean([model.wv[word] for word in response_tokens if word in model.wv], axis=0)

    return response_vector
```

- Fonction de la récupération des réponses d'entrainement selon le question_id:

```python
def find_existing_answers_vectors(df, question_id, W):

    if question_id in df['Question_id'].values:
        # Trouver tous les indices correspondants à l'ID de la question
        question_indices = df.index[df['Question_id'] == question_id].tolist()

        # Extraire les vecteurs de réponse correspondants à partir de la matrice W
        existing_answers_vectors = []
        for i in question_indices:
            existing_answers_vectors.append(W[i])
    else:
        print("L'ID de la question que vous avez fourni est incorrect")
        existing_answers_vectors = None
    existing_answers_vectors = np.array(existing_answers_vectors)
    return existing_answers_vectors
```

- Fonction de la prédiction du score en utilisant le modèle GradientBoostingRegressor :

```python
# ------------------- Model : predicting the score : -----------------------------
def prediction(response_vector, existing_answers_vectors , question_id):
    from sklearn.ensemble import GradientBoostingRegressor

    #existing_answers_vectors = find_existing_answers_vectors(df, question_id, W)

    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(existing_answers_vectors, df[(df['Question_id'] == question_id)]['Score'], test_size=0.2, random_state=42)

    # Create and train the Gradient Boosted Regression model
    gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
    gb_model.fit(X_train, y_train)

    # Reshape response vector to have shape (1, n)
    response_vector_reshaped = response_vector.reshape(1, -1)

    # Predict the score using the Gradient Boosted Regression model
    score_prediction = gb_model.predict(response_vector_reshaped)
    #print("Predicted score:", score_prediction[0])
    return score_prediction[0]
```

✓ Au niveau de PhpMyAdmin :

Table Answers :

Les réponses saisies par l'utilisateur s'enregistrent dans cette table pour calculer le score et afficher le résultat à la fin de l'examen

	id	text_answer	question_id	student_id	created_at	updated_at
Éditer Copier Supprimer	178	استيلاء الدول الأوروبية على أراضي المغرب بالقوة	1	1	2023-05-21 19:11:08	2023-05-21 19:11:08
Éditer Copier Supprimer	181	الحركة الوطنية المغربية هي حركة سياسية	4	1	2023-05-21 19:15:45	2023-05-21 19:15:45
Éditer Copier Supprimer	182	الفينيقيين	5	1	2023-05-21 19:16:00	2023-05-21 19:16:00
Éditer Copier Supprimer	185	المجموعة الأمازيغية التي عاشت في شمال إفريقيا بما ...	8	1	2023-05-21 19:16:56	2023-05-21 19:16:56
Éditer Copier Supprimer	183	المغرب يتميز بنظام حكم ديمقراطي دستوري ويضم مجلسين...	6	1	2023-05-21 19:16:20	2023-05-21 19:16:20
Éditer Copier Supprimer	180	بعد فترة طويلة من الانسحاب أصبح المغرب عضواً كاملا...	3	1	2023-05-21 19:15:07	2023-05-21 19:15:07
Éditer Copier Supprimer	187	تستخدم كلمة المغرب لوصف المنطقة الغربية لشمال إفري...	10	1	2023-05-21 19:18:21	2023-05-21 19:18:21
Éditer Copier Supprimer	184	حركة الاستقلال هي حركة سياسية نشأت في القرن العشري...	7	1	2023-05-21 19:16:41	2023-05-21 19:16:41
Éditer Copier Supprimer	179	خلال حرب الصحراء الغربية كانت العلاقة بين المغرب و...	2	1	2023-05-21 19:12:51	2023-05-21 19:12:51
Éditer Copier Supprimer	186	فُرضت الحماية الفرنسية على المغرب بعد توقيع معاهدة...	9	1	2023-05-21 19:17:22	2023-05-21 19:17:22

Table Grade :

	id	answer_id	student_id	score	created_at	updated_at
Éditer Copier Supprimer	49	178	1	5	2023-05-21 19:11:11	2023-05-21 19:11:11
Éditer Copier Supprimer	50	179	1	3	2023-05-21 19:12:51	2023-05-21 19:12:51
Éditer Copier Supprimer	51	180	1	3	2023-05-21 19:15:07	2023-05-21 19:15:07
Éditer Copier Supprimer	52	181	1	3	2023-05-21 19:15:45	2023-05-21 19:15:45
Éditer Copier Supprimer	53	182	1	1	2023-05-21 19:16:00	2023-05-21 19:16:00
Éditer Copier Supprimer	54	183	1	5	2023-05-21 19:16:20	2023-05-21 19:16:20
Éditer Copier Supprimer	55	184	1	5	2023-05-21 19:16:42	2023-05-21 19:16:42
Éditer Copier Supprimer	56	185	1	5	2023-05-21 19:16:56	2023-05-21 19:16:56
Éditer Copier Supprimer	57	186	1	4	2023-05-21 19:17:22	2023-05-21 19:17:22

Table Questions :

	text_question	id	created_at	updated_at
Éditer Copier Supprimer	ما هي العلاقة بين المغرب و الدول الأوروبية خلال ال...	1	2023-05-08 12:41:44	2023-05-08 12:41:44
Éditer Copier Supprimer	ما هي العلاقة بين المغرب و الجزائر خلال حرب الصحرا...	2	2023-05-08 12:42:24	2023-05-08 12:42:24
Éditer Copier Supprimer	ما هي علاقة المغرب بالاتحاد الأفريقي	3	2023-05-08 12:43:50	2023-05-08 12:43:50
Éditer Copier Supprimer	ما هي الحركة الوطنية في المغرب	4	2023-05-08 12:44:13	2023-05-08 12:44:13
Éditer Copier Supprimer	ما هي الحضارات التي سكنت المغرب	5	2023-05-08 12:46:02	2023-05-08 12:46:02
Éditer Copier Supprimer	ما هو نظام الحكم في المغرب	6	2023-05-08 12:46:14	2023-05-08 12:46:14
Éditer Copier Supprimer	ما هي حركة الاستقلال في المغرب	7	2023-05-08 12:46:45	2023-05-08 12:46:45
Éditer Copier Supprimer	من هم البربر	8	2023-05-08 12:46:56	2023-05-08 12:46:56
Éditer Copier Supprimer	متى فرضت الحماية الفرنسية	9	2023-05-08 12:47:30	2023-05-08 12:47:30
Éditer Copier Supprimer	ما هو معنى اسم المغرب	10	2023-05-08 12:47:45	2023-05-08 12:47:45

Table Student :

←T→	first_name	last_name	email	id	created_at	updated_at
Éditer Copier Supprimer	user	user	usere@gmail.com	1	2023-05-19 22:50:54	2023-05-19 22:50:54

VI. Frontend

i. Dépendances frontales

Comme nous allons utiliser Angular pour créer notre application frontale, nous aurons besoin de **Node.js** et **de NPM** installés sur votre machine.

```
C:\Users\hp>node --version
v16.14.2

C:\Users\hp>npm --version
9.2.0
```

Après avoir correctement installé Node.js et NPM, vous pouvez utiliser la commande npm pour installer **CLI Angular**.

On utilise cette **CLI** (Command Line Interface) pour démarrer l'application frontale, démarrer un serveur de développement et créer Angular components, services, etc .

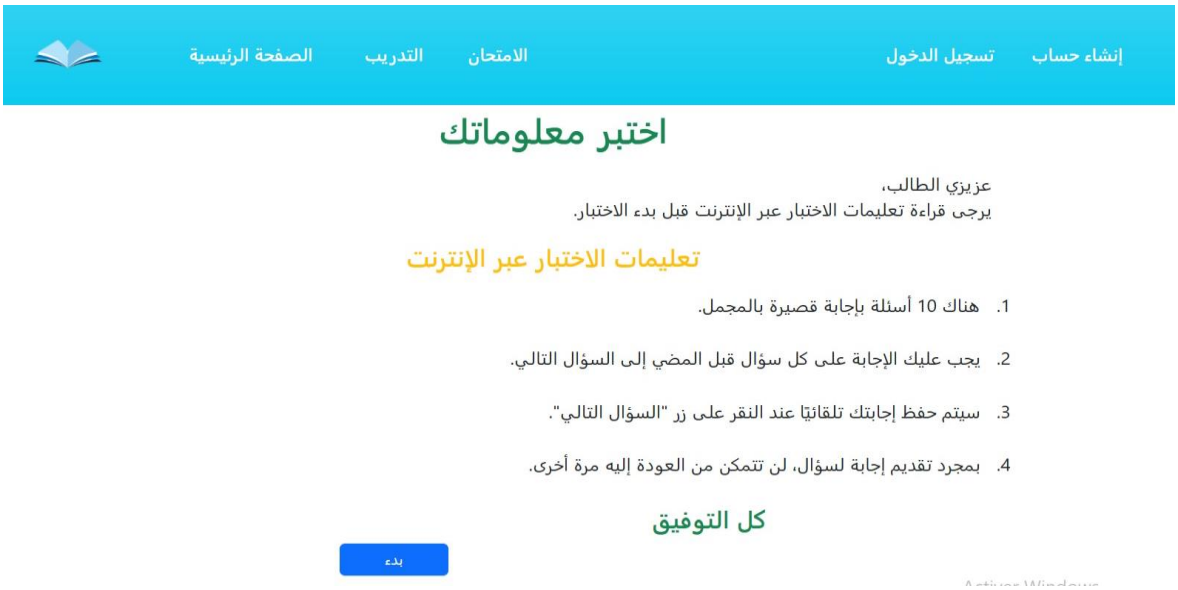
```
C:\Users\hp>ng version

     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 13.3.3
Node: 16.14.2
Package Manager: npm 9.2.0
OS: win32 x64
```

ii. Les interfaces :

✓ Page d'accueil :

✓ L'interface avant commencer l'examen :

Si l'élève a choisi de passé l'examen cette page s'affiche !



✓ Répondre aux questions :

Il faut répondre à chaque question avant de cliquer sur : la question suivante !

.....

Après répondre les 10 questions, une interface qui s’affiche pour demander de voir les réponses enregistrées avec leurs scores et le score total obtenu :

Le résultat final :

رقم السؤال	السؤال	الإجابة	الدرجة
1	ما هي العلاقة بين المغرب و الدول الأوروبية خلال القرن التاسع عشر	استيلاء الدول الأوروبية على أراضي المغرب بالقوة	5
2	ما هي العلاقة بين المغرب و الجزائر خلال حرب الصحراء الغربية	خلال حرب الصحراء الغربية كانت العلاقة بين المغرب والجزائر متوترة ومعقدة للغاية	3
3	ما هي علاقة المغرب بالاتحاد الأفريقي	بعد فترة طويلة من الانسحاب أصبح المغرب عضواً كاملاً في الاتحاد الأفريقي	3
4	ما هي الحركة الوطنية في المغرب	الحركة الوطنية المغربية هي حركة سياسية	3
5	ما هي الحضارات التي سكنت المغرب	الفينيقيين	1
6	ما هو نظام الحكم في المغرب	المغرب يتميز بنظام حكم ديمقراطي دستوري ويضم مجلسين تشريعيين هما مجلس المستشارين ومجلس النواب وهو ما يساعد على تحقيق العدالة والتنمية في البلاد	5
7	ما هي حركة الاستقلال في المغرب	حركة الاستقلال هي حركة سياسية نشأت في القرن العشرين و تهدف إلى الحصول على الاستقلال التام للمغرب	5
8	من هم البربر	المجموعة الأمازيغية التي عاشت في شمال إفريقيا بما في ذلك المغرب تعرف باسم البربر	5
9	متى فرضت الحماية الفرنسية	فُرضت الحماية الفرنسية على المغرب بعد توقيع معاهدة الحماية من قبل السلطان عبد الحفيظ في 1 مارس	4
10	ما هو معنى اسم المغرب	تستخدم كلمة المغرب لوصف المنطقة الغربية لشمال إفريقيا	2

VII. Conclusion

En conclusion, ce projet de machine learning a été un succès complet, depuis la création du modèle jusqu'à son déploiement. Nous avons réussi à créer un modèle de prédiction précis qui est capable de valoriser les réponses efficacement. Nous avons travaillé avec diligence pour préparer les données, nettoyer et traiter les données, entraîner et affiner notre modèle, et évaluer ses performances.

En fin de compte, ce projet nous a permis de développer notre compréhension des concepts clés du machine learning, de renforcer nos compétences en programmation (notamment les deux intéressants frameworks Flask et Angular) et en analyse de données, et de nous familiariser avec des nouveaux outils et frameworks.

En somme, ce projet a été une expérience enrichissante et gratifiante pour nous, et nous sommes fiers de notre travail accompli. Nous avons démontré notre capacité à travailler efficacement en équipe, à relever des défis complexes et à livrer un produit de qualité qui répond aux besoins de l'utilisateur final.

VIII. Bibliographique

https://auth0.com/blog/using-python-flask-and-angular-to-build-modern-apps-part-1/#Bootstrapping-the-Angular-Application

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0272269

https://github.com/bz866/Automated-Scoring-System

https://ieeexplore.ieee.org/document/10102440

https://mawdoo3.com/%D9%85%D8%B9%D9%84%D9%88%D9%85%D8%A7%D8%AA_%D8%B9%D9%86_%D8%AA%D8%A7%D8%B1%D9%8A%D8%AE_%D8%A7%D9%84%D9%85%D8%BA%D8%B1%D8%A8

https://tipz.io/redir?user_type=12&type=sr&redir=eJzLKCkpKLbS1y8vL9fLykrOLNHLL0rXT8kvz8vJT0xJy8xJ1TczZGAwNLMwMTO0NDC3ZHhSPn_-R2VfGfWcL2u32-xwBQANLxfz&src=93bfda&via_page=1

https://blog.filkhabr.com/p/%D8%A3%D8%B3%D8%A6%D9%84%D8%A9-%D8%AB%D9%82%D8%A7%D9%81%D9%8A%D8%A9-%D8%AD%D9%88%D9%84-%D8%AA%D8%A7%D8%B1%D9%8A%D8%AE-%D8%A7%D9%84%D9%85%D8%BA%D8%B1%D8%A8-%D8%A3%D8%B3%D8%A6%D9%84%D8%A9-%D8%AB%D9%82%D8%A7%D9%81%D9%8A%D8%A9-%D8%B9%D8%A7%D9%85%D8%A9-%D9%85%D8%B9-%D8%A7%D9%84%D8%A5%D8%AC%D8%A7%D8%A8%D8%A9

https://ar.e-leath.net/%d8%a7%d8%b3%d8%a6%d9%84%d8%a9-%d8%aa%d8%a7%d8%b1%d9%8a%d8%ae%d9%8a%d8%a9-%d8%b9%d9%86-%d8%a7%d9%84%d9%85%d8%ba%d8%b1%d8%a8/

https://fr.scribd.com/document/518279268/300-%D8%B3%D8%A4%D8%A7%D9%84-%D9%88%D8%AC%D9%88%D8%A7%D8%A8-%D8%AD%D9%88%D9%84-%D8%AA%D8%A7%D8%B1%D9%8A%D8%AE-%D8%A7%D9%84%D9%85%D8%BA%D8%B1%D8%A8