

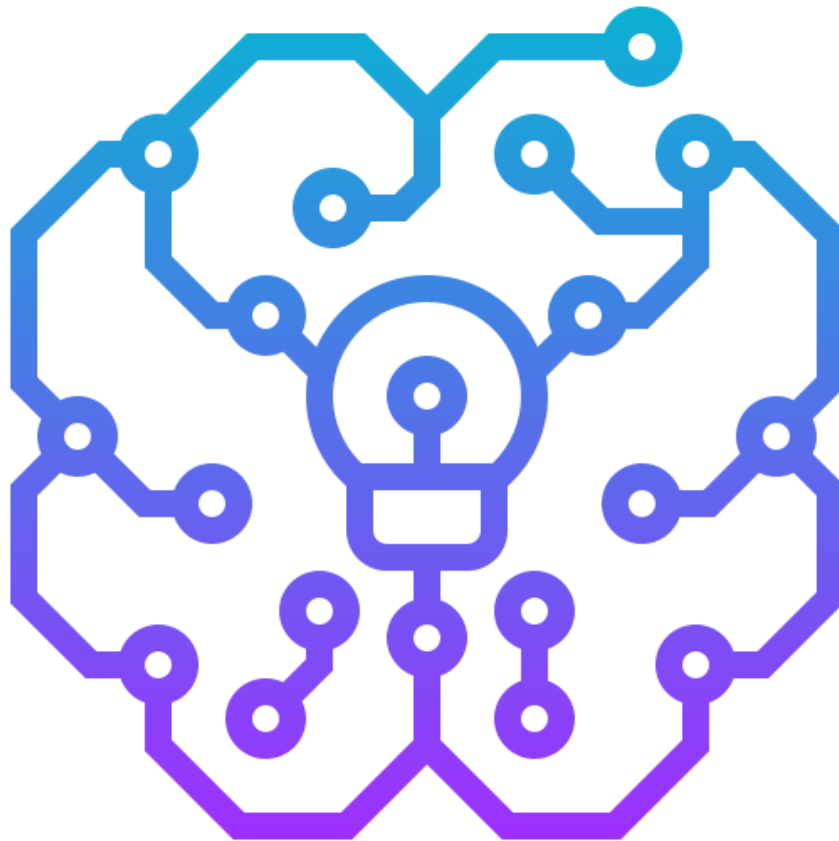


كلية العلوم والتقنيات بطنجة  
Faculté des Sciences et Techniques de Tanger



## Machine Learning Project LSI 2023

### Project 1: Arabic Automated short answers grading system for Moroccan history



Réalise par :

EL KRISSI Achraf  
AKDI Nafia

Encadre par :

Mr. EL AACHAK Lotfi

# Table des matières

<b>INTRODUCTION.....</b>	<b>3</b>
○ CRISP	
<b>LA COMPRÉHENSION DU PROBLÈME MÉTIER.....</b>	<b>4</b>
<b>LA COMPRÉHENSION DES DONNÉES.....</b>	<b>5</b>
<b>LA CONSTRUCTION DU DATA HUB (PRÉPARATION DES DONNÉES)...</b>	<b>8</b>
<b>LA MODÉLISATION ET EVALUATION DES MODÈLES .....</b>	<b>13</b>
○ MODÉLISATION	
○ EVALUATION	
○ RÉSUMÉ	
○ LA SAUVEGARDE DES MODÈLES	
<b>LE DÉPLOIEMENT DU MODÈLE.....</b>	<b>24</b>
○ FastAPI	
○ Angular	
○ GraphQL	
<b>MISE EN CONTENEUR AVEC DOCKER.....</b>	<b>29</b>
<b>APPLICATION.....</b>	<b>33</b>

## Introduction :

Le domaine de l'apprentissage automatique et de l'exploration de données offre des opportunités passionnantes pour extraire des connaissances précieuses à partir de vastes ensembles de données. Cependant, pour réussir ces projets, il est essentiel d'adopter une approche structurée et méthodique. C'est là que la méthodologie **CRISP (Cross Industry Standard Process for Data Mining)** entre en jeu.

La méthodologie CRISP fournit un cadre détaillé et itératif pour mener à bien des projets d'exploration de données et de modélisation prédictive. Elle est largement utilisée dans l'industrie et les domaines de recherche pour garantir une approche cohérente tout au long du processus. Dans ce rapport, nous allons explorer les différentes étapes de la méthodologie CRISP et les appliquer à notre projet de machine learning.

La première étape de la méthodologie CRISP consiste à comprendre le problème et les objectifs. Il est essentiel de définir clairement le problème à résoudre et d'identifier les objectifs commerciaux ou opérationnels sous-jacents. Cela nous permettra de cibler nos efforts de modélisation et de fournir des résultats pertinents pour la prise de décision.

La deuxième étape est la compréhension des données. Nous devons collecter les données pertinentes pour notre projet et les explorer en détail. Cela nous permettra de comprendre la qualité, le format et la structure des données, ainsi que d'identifier d'éventuels problèmes ou erreurs.

La troisième étape est la préparation des données. Cette étape implique le nettoyage, la transformation et la préparation des données pour l'analyse ultérieure. Nous devons traiter les valeurs manquantes, les données aberrantes et les variables catégorielles, et

normaliser les données si nécessaire. Une préparation minutieuse des données est essentielle pour garantir des résultats fiables et précis.

La quatrième étape est la modélisation des données. Nous appliquerons des techniques d'apprentissage automatique et de modélisation prédictive pour construire des modèles statistiques. Selon les objectifs de notre projet, nous pouvons utiliser des méthodes telles que la régression, la classification ou le clustering. Nous évaluerons également la performance de nos modèles pour sélectionner le meilleur modèle possible.

dans la cinquième étape, nous évaluerons les de notre modèle. Nous examinerons les métriques d'évaluation appropriées pour évaluer la performance de notre modèle. Si les résultats sont satisfaisants, nous pourrions déployer le modèle en production pour une utilisation continue.

Le déploiement est l'étape finale du processus. Elle consiste en une mise en production pour les utilisateurs finaux des modèles obtenus. Son objectif : mettre la connaissance obtenue par la modélisation, dans une forme adaptée, et l'intégrer au processus de prise de décision.

Le déploiement peut ainsi aller, selon les objectifs, de la simple génération d'un rapport décrivant les connaissances obtenues jusqu'à la mise en place d'une application, permettant l'utilisation du modèle obtenu, pour la prédiction de valeurs inconnues d'un élément d'intérêt.

### **La compréhension du problème métier :**

Dans le cadre de ce projet, notre objectif est de développer un système automatisé de notation des réponses courtes en arabe pour l'histoire marocaine. L'objectif principal est de fournir une évaluation précise et adéquate aux étudiants en fonction de leurs réponses. Nous souhaitons évaluer leur compréhension des concepts

historiques, leur capacité à analyser les informations pertinentes et à formuler des arguments cohérents.

Grâce à ce système, nous visons à améliorer le processus de notation des réponses courtes en offrant une solution automatisée qui permettra aux enseignants de gagner du temps et de fournir des commentaires plus précis et constructifs aux étudiants. Nous cherchons à créer un environnement d'apprentissage plus efficace et interactif, où les étudiants pourront recevoir un retour rapide sur leurs performances et disposer d'un outil qui les aidera à comprendre leurs forces et leurs faiblesses dans la matière.

En comprenant pleinement le problème métier, nous serons en mesure de développer un système qui répondra aux besoins spécifiques des enseignants et des étudiants. Nous chercherons à fournir une évaluation objective et cohérente, tout en tenant compte des particularités linguistiques et culturelles propres à l'arabe et à l'histoire marocaine.

En résumé, notre objectif est de développer un système automatisé de notation des réponses courtes en arabe pour l'histoire marocaine qui fournira une évaluation précise et équitable aux étudiants, tout en offrant aux enseignants un outil efficace pour évaluer les performances des étudiants et leur fournir des commentaires constructifs.

### La compréhension des données

Dans ce projet, nous avons mis en place un processus pour évaluer les élèves à travers 10 questions portant sur l'histoire du Maroc. Ces questions ont été soigneusement sélectionnées par un **ancien**

enseignant du lycée Hassan II à Tanger, afin de garantir leur pertinence et leur adéquation avec le programme d'études.

Pour collecter les réponses des élèves, nous avons créé un formulaire en ligne ([aller au formulaire](#)).

## أسئلة حول تاريخ المغرب

**ملاحظات مهمة :**

- المرجو الإجابة على هذه الأسئلة باللغة العربية .
- المرجو الجواب اعتمادا على مكتبياتكم الدراسية و ليس عبر الانترنت .
- المرجو صياغة الاجوبة بأسلوبك الخاص
- لإرسال الاجوبة يجب أن تكون متصل بشبكة الانترنت (\*3 او WIFI و ليس \*6 )
- في حال حدوث اي مشكل المرجو التواصل معنا على الرقم التالي [0695558174](tel:0695558174)

**- تحقق من أنك قمت بتسجيل الدخول باستخدام البريد الإلكتروني لحفظ الاجوبة**

elkrissiaxraf701@gmail.com [Changer de compte](#)

Non partagé

**\* Indique une question obligatoire**

**\* سؤال 1 :** ما هي ظروف فرض الحماية الفرنسية على المغرب ؟

Votre réponse

**\* سؤال 2 :** ما هي مراحل الاحتلال العسكري للمغرب ؟

Votre réponse

**\* سؤال 3 :** اذكر اسماء اهم المقاومين المغاربة و بين دورهم في مواجهة الاحتلال ؟

Votre réponse

**\* سؤال 4 :** ما مظاهر الاستغلال الاستعماري للمغرب ؟

Votre réponse

\* سؤال 5 : ما هي اتصالات الاستغلال الاستعماري على الاقتصاد و المجتمع المغربيين ؟

Votre réponse

\* سؤال 6 : ما هي ظروف ظهور الحركة الوطنية ؟

Votre réponse

\* سؤال 7 : ما هي ظروف تقديم المغرب لوثيقة الاستقلال ؟

Votre réponse

\* سؤال 8 : ما علاقة ثورة الملك و الشعب باستقلال المغرب؟

Votre réponse

\* سؤال 9 : ما هي الضغوط العسكرية التي استعملتها الدول الامبريالية للتدخل الاوروبي في المغرب خلال القرن 19 م ؟

Votre réponse

\* سؤال 10 : ما نوعية الاصلاحات التي باشرها المغرب خلال القرن 19 م و ما مآلها؟

Votre réponse

Envoyer

Effacer le formulaire

Ce formulaire permet aux élèves de répondre aux questions de manière structurée et organisée. Chaque réponse est automatiquement enregistrée dans un fichier Excel dédié, ce qui facilite la gestion et l'analyse des données.

Nous avons également essayé d'évaluer la qualité des réponses en se basant sur la réponse parfaite donnée par l'enseignant . Chaque réponse reçoit un score en fonction de sa précision, de sa pertinence et de la qualité de son argumentation. **Un score de 0** est attribué si la réponse est incorrecte, **un score de 1** est donné si la réponse est

correcte mais manque d'argumentation solide, et un score de 2 est attribué si la réponse est parfaite, avec une argumentation claire et convaincante.

A		B	C
number	value		score
2	1	ظروف خارجية : التنافس الإمبريالي على المغرب نتج عنه توقيع اتفاقات انتهت بآنفراد فرنسا وإسبانيا باحتلال المغرب الاتفاق الفرنسي الإيطالي 1902 حول المغرب وليبيا والاتفاق الألماني 1911 حول مة الظروف التي أدت إلى فرض الحماية الفرنسية على المغرب تشمل عوامل خارجية مثل المنافسة الإمبريالية بين القوى الأوروبية ، والتي أدت إلى توقيع اتفاقيات منحت فرنسا وإسبانيا احتكار احتلال المغرب. وتند	2
2	1	يرجع فرض الحماية الفرنسية على المغرب إلى عدة أسباب، منها الظروف الخارجية كالتنافس الإمبريالي بين فرنسا وإسبانيا وتوقيع اتفاقات احتلال المغرب، والاتفاق الفرنسي الإيطالي عام 1902 حول المغرب وليبيا والاتفاق الفرنسي الألماني في 1911 حول المغرب والكونغو، وكذلك الظروف الداخلية مثل توالي السلاطين وتزايد عدد الثورات والتمردات، وتزايد حجم القروض الأجنبية وإلحاق كاهل الفلاحين بالضرائب، وانتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	إن فرض الحماية الفرنسية على المغرب يعود إلى عدة أسباب، منها الأسباب الخارجية التي تتمثل في التنافس الإمبريالي بين فرنسا وإسبانيا وتوقيع اتفاقات لاحتلال المغرب، بما في ذلك الاتفاق الفرنسي الإيطالي 1902 فرض الحماية الفرنسية على المغرب يعود إلى عدة أسباب، بما في ذلك الضغوطات الخارجية مثل التنافس الإمبريالي بين فرنسا وإسبانيا والاتفاقات التي أدت إلى احتلال المغرب، وكذلك الظروف الداخلية مثل توالي السلاطين وتزايد الثورات والتمردات وانتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	ترجع أسباب فرض الحماية الفرنسية على المغرب إلى مجموعة من العوامل، بما في ذلك التنافس الإمبريالي بين فرنسا وإسبانيا والاتفاقات التي أدت إلى احتلال المغرب، وكذلك التحديات الداخلية مثل توالي السلاطين وتزايد الثورات والتمردات وانتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	يرجع فرض الحماية الفرنسية على المغرب إلى عدة عوامل، منها التنافس الإمبريالي بين فرنسا وإسبانيا وتوقيع اتفاقات التي أدت إلى احتلال المغرب، والتحديات الداخلية مثل توالي السلاطين وتزايد الثورات والتمردات وانتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	عدة أسباب دفعت فرنسا لفرض الحماية على المغرب، ومنها الظروف الخارجية كالتنافس الإمبريالي مع إسبانيا وتوقيع اتفاقات احتلال المغرب بما في ذلك الاتفاق الفرنسي الإيطالي لعام 1902 حول المغرب وليبيا، والاتفاق الفرنسي الألماني لعام 1911 حول المغرب والكونغو. ومن الأسباب الداخلية، تزايد عدد الثورات والتمردات وتوالي السلاطين وانتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	توقيع اتفاقات بالاحتلال الفرنسي والإسباني للمغرب كان نتيجة للتنافس الإمبريالي على المغرب، وتم توقيع اتفاقا الفرنسي الإيطالي في عام 1902 بشأن المغرب وليبيا، واتفقت فرنسا مع ألمانيا في عام 1911 على المغرب والكونغو. ومن الجوانب الداخلية، ضعفت السلطة المركزية بعد وفاة حاجب السلطان باحماد في عام 1900، وزادت عدد الثورات والتمردات بسرعة نتيجة تعاقب السلاطين، مثل تمرد الريسوني والجيلالي الزرعوني. وزاد حجم القروض الأجنبية، وتحمل الفلاحون الكثير من الضرائب والأعباء، مما أدى إلى انتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	الاحتلال الفرنسي والإسباني للمغرب جاء نتيجة للتنافس الإمبريالي بين القوى العظمى في ذلك الوقت، وتم توقيع اتفاقا الفرنسي الإيطالي في عام 1902 بشأن المغرب وليبيا، كما تم الاتفاق بين فرنسا وألمانيا على المغرب والكونغو في عام 1911. ومن الناحية الداخلية، تركزت السلطة المركزية بضعف بعد وفاة حاجب السلطان بلمعد في عام 1900، وشهدت البلاد عداً كبيراً بين الثورات والتمردات بسبب تعاقب السلاطين بسرعة، مثل تمرد الريسوني والجيلالي الزرعوني. وتزايد حجم القروض الأجنبية، وحمل الفلاحون الكثير من الضرائب والأعباء، مما أدى إلى انتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2
2	1	التنافس الإمبريالي بين فرنسا وإسبانيا كان من بين الأسباب التي أدت إلى فرض الحماية الفرنسية على المغرب وتوقيع اتفاقات احتلال المغرب، بما في ذلك الاتفاق الفرنسي الإيطالي عام 1902 حول المغرب وليبيا والاتفاق الفرنسي الألماني في 1911 حول المغرب والكونغو. ومن الجوانب الداخلية، تزايدت الثورات والتمردات في ظل توالي السلاطين، مثل تمرد الريسوني والجيلالي الزرعوني، وتزايد حجم القروض الأجنبية والضرائب الثقيلة التي حملت الفلاحين الكثير منها، مما أدى إلى انتشار الفقر والأمراض والأوبئة والبطالة والمجاعة.	2

## La construction du Data Hub (préparation des données) :



```
# read csv file into pandas
df = pd.read_excel("quest10.xlsx")
df.head(10)
```

	number	value	score
0	10.0	أقام المغرب مدارس جديدة لتعليم الفتيات	2
1	10.0	...المغرب باشر إصلاحات اجتماعية لتحسين الوضع الاق	1
2	10.0	...المغرب لم يقم بأي تغييرات اجتماعية يستحق الذكر	0
3	10.0	قام المغرب بإنشاء مستشفيات لتحسين الوضع الصحي	2
4	10.0	فتح المغرب بعض المدارس لتعليم العلوم الحديثة	1
5	10.0	... المغرب لم يقم بأي إصلاحات اجتماعية خلال القرن	0
6	10.0	... أنشأ المغرب بيوت الأيتام ودور الرعاية للفقراء	2
7	10.0	نشر المغرب الوعي الصحي والتثقيف الصحي	1
8	10.0	...المغرب لم يقم بأي تغييرات اجتماعية يستحق الذكر	0
9	10.0	...عمل المغرب على تحسين حماية حقوق العمال ونظام ا	2

```
# infos sur notre dataset
df.info()
# Les valeurs manquantes
df.isnull().sum()
# On remarque qu'il n'y a pas de valeurs nulles
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97 entries, 0 to 96
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0    number    97 non-null      int64
1    value     97 non-null      object
2    score     97 non-null      int64
dtypes: int64(2), object(1)
memory usage: 2.4+ KB

number    0
value     0
score     0
dtype: int64
```

```
# les valeurs dupliquées
df.duplicated().sum()
```

0

```

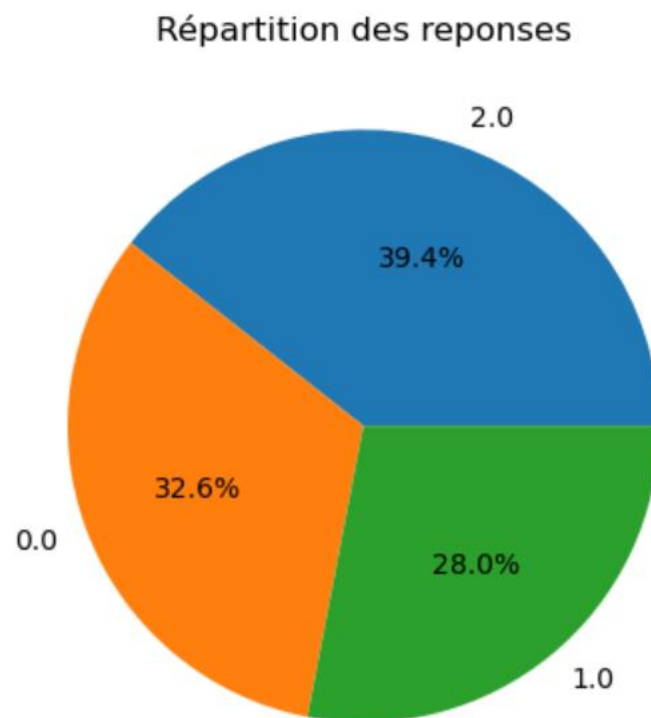
# Visualisation
score_counts = df['score'].value_counts()

# Créer Le pie chart pour La répartition des scores
plt.pie(score_counts, labels=score_counts.index, autopct='%1.1f%%')

# Ajouter un titre
plt.title('Répartition des reponses')

# Afficher Le graphique
plt.show()

```



```

score_counts = df['score'].value_counts()

# Créer une liste de couleurs pour chaque score
colors = ['#ff9999', '#66b3ff', '#99ff99']

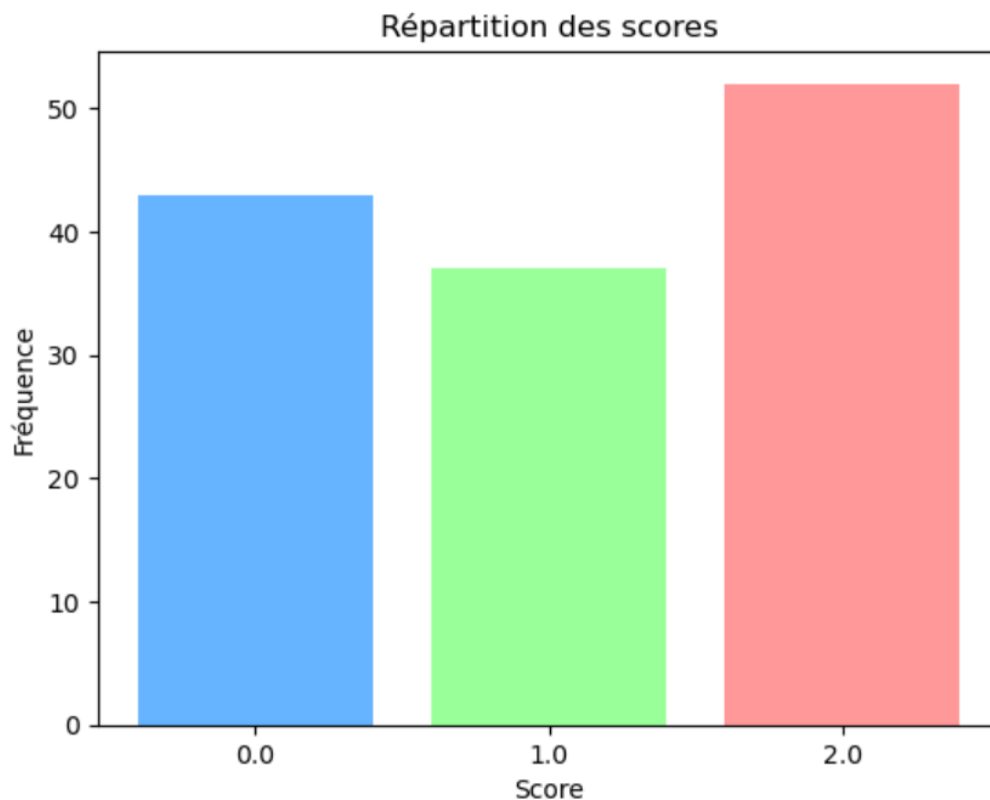
# Créer Le bar chart avec des couleurs personnalisées
plt.bar(score_counts.index, score_counts.values, color=colors)

# Ajouter des étiquettes pour chaque valeur du score
plt.xticks(score_counts.index, score_counts.index)

# Ajouter un titre et des labels d'axe
plt.title('Répartition des scores')
plt.xlabel('Score')
plt.ylabel('Fréquence')

# Afficher Le graphique
plt.show()

```



On peut dire que notre data est balancé (**données équilibrées**)

```
stop_words = set(stopwords.words('arabic'))
def preprocess_text(text):
    text = text.lower()
    tokens = word_tokenize(text)
    #tokens = [token for token in tokens if token.isalpha() and token not in stop_words]
    return tokens

df['tokens'] = df['value'].apply(preprocess_text)
```

df

	number	value	score	tokens
0	10.0	أقام المغرب مدارس جديدة لتعليم الفتيات	2	[أقام, المغرب, مدارس, جديدة, لتعليم, الفتيات]
1	10.0	المغرب باشر إصلاحات اجتماعية لتحسين الوضع الاق...	1	...المغرب, باشر, إصلاحات, اجتماعية, لتحسين, الوضع]
2	10.0	المغرب لم يقم بأي تغييرات اجتماعية يستحق الذكر	0	...المغرب, لم, يقم, بأي, تغييرات, اجتماعية, يستحق]
3	10.0	قام المغرب بإنشاء مستشفيات لتحسين الوضع الصحي	2	...قام, المغرب, بإنشاء, مستشفيات, لتحسين, الوضع]
4	10.0	فتح المغرب بعض المدارس لتعليم العلوم الحديثة	1	...فتح, المغرب, بعض, المدارس, لتعليم, العلوم, ال]
...	...	...	...	...
df	...	...	...	...

Une étape essentielle dans le prétraitement des données textuelles est d'appliquer des transformations pour rendre le texte plus adapté à l'analyse. La fonction 'preprocess\_text()' a été développée pour effectuer une série de prétraitements sur le texte (Tokenisation)

```
model = Word2Vec(df['tokens'], vector_size=100, window=5, min_count=1, workers=4)
```

```
def get_feature_vector(tokens, model):
    feature_vector = np.zeros((100,), dtype='float32')
    for token in tokens:
        try:
            feature_vector += model.wv[token]
        except KeyError:
            continue
    return feature_vector

df['feature_vector'] = df['tokens'].apply(lambda x: get_feature_vector(x, model))
df['feature_vector']
```

```
0    [-0.021790393, 0.018973054, 0.007494045, 0.017...
1    [0.010273369, 0.0039946227, -0.015632702, -0.0...
2    [-0.027587626, 0.057505667, -0.055178463, -0.0...
3    [-0.06314353, 0.06384929, 0.0042380835, 0.0497...
4    [-0.018355379, 0.0036655972, -0.015970552, 0.0...
```

Une autre étape clé dans le prétraitement des données textuelles consiste à représenter les mots sous forme de vecteurs numériques, afin de pouvoir les utiliser comme entrées dans les algorithmes d'apprentissage automatique. La fonction 'get\_feature\_vector()' a été développée pour obtenir les vecteurs de caractéristiques correspondants à une liste de tokens donnée.

Cette fonction prend en entrée une liste de tokens, représentant les mots prétraités d'un texte, ainsi qu'un modèle de vectorisation des mots préalablement entraîné (un modèle Word2Vec). Elle initialise un vecteur de caractéristiques de dimension fixe (de taille 100) avec des valeurs nulles.

Ensuite, pour chaque token dans la liste, la fonction vérifie si le modèle de vectorisation contient une représentation vectorielle correspondante pour ce token. Si c'est le cas, le vecteur de caractéristiques est mis à jour en ajoutant le vecteur du token au vecteur existant.

Finalement, la fonction renvoie le vecteur de caractéristiques obtenu, qui représente la somme des vecteurs des tokens présents dans le texte d'origine.

## **la modélisation des données et Evaluation :**

### **➤ modélisation des données :**

```
train_data = df.sample(frac=0.80, random_state=42)
test_data = df.drop(train_data.index)
```

```
X_train = np.array(train_data['feature_vector'].tolist())
Y_train = np.array(train_data['score'])
```

```
X_test = np.array(test_data['feature_vector'].tolist())
Y_test = np.array(test_data['score'])
```

nous effectuons une division des données en un ensemble d'entraînement (train\_data) et un ensemble de test (test\_data) pour évaluer les performances de notre modèle.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# KNN
def KNN():
    KNN = KNeighborsClassifier(n_neighbors=5)
    KNN.fit(X_train, Y_train)
    return KNN
# Decision Tree
def DT():
    DT = DecisionTreeClassifier()
    DT.fit(X_train, Y_train)
    return DT
# ANN
def ANN():
    ANN = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
    ANN.fit(X_train, Y_train)
    return ANN
# NB
def NB():
    NB = GaussianNB()
    NB.fit(X_train, Y_train)
    return NB
# SVM
def SVM_RBF():
    SVM = SVC(kernel='rbf', probability=True)
    SVM.fit(X_train, Y_train)
    return SVM
def SVM_LINEAR():
    SVM = SVC(kernel='linear', probability=True)
    SVM.fit(X_train, Y_train)
    return SVM
def SVM_POLY():
    SVM = SVC(kernel='poly', probability=True)
    SVM.fit(X_train, Y_train)
    return SVM
```

nous définissons plusieurs fonctions qui correspondent à différents algorithmes de classification que nous allons utiliser pour entraîner des modèles sur nos données.

```
#-----entraînement-----
KNN= KNN()
DT = DT()
NB=NB()
ANN=ANN()
SVM_RBF=SVM_RBF()
SVM_LINEAR=SVM_LINEAR()
SVM_POLY=SVM_POLY()
```

Nous appelons chaque fonction correspondant à un algorithme de classification (KNN, DT, NB, ANN, SVM\_RBF, SVM\_LINEAR, SVM\_POLY) pour obtenir les modèles entraînés respectifs.

**Remarque : ces étapes sont répétées pour chaque question c a d on aura 10 modèles**

### ➤ Evaluation des 10 modèles :

- modèle 1 (question 1) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
```

```
Accuracy KNN : 94.44%
Accuracy DT : 83.33%
Accuracy ANN : 83.33%
Accuracy NB : 72.22%
Accuracy SVM_RBF : 83.33%
Accuracy SVM_LINEAR : 83.33%
Accuracy SVM_POLY : 88.89%
```

```
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.986441798941799
AUC-ROC DT score: 0.8672619047619047
AUC-ROC ANN score: 0.9100529100529101
AUC-ROC NB score: 0.8998677248677248
AUC-ROC SVM_RBF score: 0.9412037037037037
AUC-ROC SVM_LINEAR score: 0.8971560846560847
AUC-ROC SVM_POLY score: 0.9375
```

```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
#print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.8028571428571428
Mean score SVM LINEAR: 0.6904761904761905
Mean score SVM POLY: 0.6171428571428571
Mean score KNN: 0.7876190476190476
```

## • modèle 2 (question 2) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
```

```
Accuracy KNN : 89.47%
Accuracy DT : 63.16%
Accuracy NB : 89.47%
Accuracy SVM_RBF : 89.47%
Accuracy SVM_LINEAR : 57.89%
Accuracy SVM_POLY : 57.89%
```

```
y_scores_KNN = KNN.predict_proba(X_test)
y_scores_DT = DT.predict_proba(X_test)
y_scores_ANN = ANN.predict_proba(X_test)
y_scores_NB = NB.predict_proba(X_test)
y_scores_SVM_RBF = SVM_RBF.predict_proba(X_test)
y_scores_SVM_LINEAR = SVM_LINEAR.predict_proba(X_test)
y_scores_SVM_POLY = SVM_POLY.predict_proba(X_test)
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.9757575757575757
AUC-ROC DT score: 0.7236652236652237
AUC-ROC NB score: 0.9813131313131312
AUC-ROC SVM_RBF score: 0.9906565656565657
AUC-ROC SVM_LINEAR score: 0.7838383838383839
AUC-ROC SVM_POLY score: 0.7651515151515151
```

```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.9233333333333333
Mean score ANN: 0.885
Mean score NB : 0.8066666666666666
Mean score SVM LINEAR: 0.6266666666666667
Mean score KNN: 0.7808333333333334
Mean score SVM POLY: 0.6266666666666667
Mean score DT: <built-in method mean of numpy.ndarray object at 0x0000029CCE4198D0>
```



- modèle 3 (question 3) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))

Accuracy KNN : 77.78%
Accuracy DT : 66.67%
Accuracy ANN : 94.44%
Accuracy NB : 83.33%
Accuracy SVM_RBF : 94.44%
Accuracy SVM_LINEAR : 88.89%
Accuracy SVM_POLY : 55.56%

print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM_LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM_POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())

Mean score RBF: 0.8142857142857144
Mean score SVM_LINEAR: 0.7333333333333333
Mean score SVM_POLY: 0.6761904761904762
Mean score ANN: 0.9285714285714285
Mean score NB : 0.7723809523809524
Mean score KNN: 0.818095238095238

y_scores_KNN = KNN.predict_proba(X_test)
y_scores_DT = DT.predict_proba(X_test)
y_scores_ANN = ANN.predict_proba(X_test)
y_scores_NB = NB.predict_proba(X_test)
y_scores_SVM_RBF = SVM_RBF.predict_proba(X_test)
y_scores_SVM_LINEAR = SVM_LINEAR.predict_proba(X_test)
y_scores_SVM_POLY = SVM_POLY.predict_proba(X_test)
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))

AUC-ROC KNN score: 0.8913690476190476
AUC-ROC DT score: 0.7779761904761905
AUC-ROC NB score: 0.9696428571428571
AUC-ROC SVM_RBF score: 0.9898809523809525
AUC-ROC SVM_LINEAR score: 0.9898809523809525
AUC-ROC SVM_POLY score: 0.6964285714285715
AUC-ROC ANN score: 1.0
```

- modèle 4 (question 4) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
```

```
Accuracy KNN : 62.96%
Accuracy DT : 51.85%
Accuracy ANN : 48.15%
Accuracy NB : 81.48%
Accuracy SVM_RBF : 85.19%
Accuracy SVM_LINEAR : 29.63%
Accuracy SVM_POLY : 66.67%
```

```
y_scores_KNN = KNN.predict_proba(X_test)
y_scores_DT = DT.predict_proba(X_test)
y_scores_ANN = ANN.predict_proba(X_test)
y_scores_NB = NB.predict_proba(X_test)
y_scores_SVM_RBF = SVM_RBF.predict_proba(X_test)
y_scores_SVM_LINEAR = SVM_LINEAR.predict_proba(X_test)
y_scores_SVM_POLY = SVM_POLY.predict_proba(X_test)
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.7867864104706209
AUC-ROC DT score: 0.6257414369256474
AUC-ROC ANN score: 0.7585700362016152
AUC-ROC NB score: 0.8995265942634364
AUC-ROC SVM_RBF score: 0.9328947368421052
AUC-ROC SVM_LINEAR score: 0.8546365914786967
AUC-ROC SVM_POLY score: 0.817000835421888
```

```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
#print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.6871794871794872
Mean score SVM LINEAR: 0.358974358974359
Mean score SVM POLY: 0.5141025641025642
Mean score ANN: 0.6269230769230769
Mean score NB : 0.5923076923076923
Mean score KNN: 0.5282051282051282
```

## • modèle 5 (question) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
```

```
Accuracy KNN : 70.00%
Accuracy DT : 55.00%
Accuracy ANN : 80.00%
Accuracy NB : 65.00%
Accuracy SVM_RBF : 65.00%
Accuracy SVM_LINEAR : 55.00%
Accuracy SVM_POLY : 70.00%
```

```

print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
#print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())

```

```

Mean score SVM POLY: 0.558102766798419
Mean score ANN: 0.6632411067193675
Mean score NB : 0.6640316205533596
Mean score KNN: 0.6209486166007905
Mean score RBF: 0.6810276679841897
Mean score SVM LINEAR: 0.574703557312253

```

## - modèle 6 (question 6) :

```

## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))

```

```

Accuracy KNN : 90.62%
Accuracy DT : 71.88%
Accuracy ANN : 81.25%
Accuracy NB : 87.50%
Accuracy SVM_RBF : 78.12%
Accuracy SVM_LINEAR : 50.00%
Accuracy SVM_POLY : 50.00%

```

```

print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
#print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())

```

```

Mean score SVM LINEAR: 0.6933333333333334
Mean score SVM POLY: 0.6799999999999999
Mean score ANN: 0.9066666666666666
Mean score NB : 0.9066666666666666
Mean score RBF: 0.7866666666666667
Mean score KNN: 0.9066666666666666

```

```

# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))

```

```

AUC-ROC KNN score: 0.9617075517075517
AUC-ROC DT score: 0.8034343434343435
AUC-ROC ANN score: 0.9896151996151996
AUC-ROC NB score: 0.9742568542568543
AUC-ROC SVM_RBF score: 0.9894949494949495
AUC-ROC SVM_LINEAR score: 0.9832611832611833
AUC-ROC SVM_POLY score: 0.9606060606060606

```

- modèle 7 (question 7) :

```
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
```

```
Accuracy KNN : 87.50%
Accuracy DT : 81.25%
Accuracy ANN : 93.75%
Accuracy NB : 87.50%
Accuracy SVM_RBF : 87.50%
Accuracy SVM_LINEAR : 50.00%
Accuracy SVM_POLY : 50.00%
```

```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())

print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.9064102564102564
Mean score SVM LINEAR: 0.6205128205128204
Mean score SVM POLY: 0.6358974358974359
Mean score ANN: 0.8910256410256409
Mean score NB : 0.8589743589743591
Mean score KNN: 0.7448717948717949
```

# Affichage du score AUC-ROC

```
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.9453125
AUC-ROC DT score: 0.8402777777777778
AUC-ROC ANN score: 1.0
AUC-ROC NB score: 1.0
AUC-ROC SVM_RBF score: 1.0
AUC-ROC SVM_LINEAR score: 0.7100694444444445
AUC-ROC SVM_POLY score: 0.6493055555555555
```

- modèle 8 (question 8) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
```

```
Accuracy KNN : 82.76%
Accuracy DT : 65.52%
Accuracy ANN : 86.21%
Accuracy NB : 79.31%
Accuracy SVM_RBF : 89.66%
Accuracy SVM_LINEAR : 68.97%
Accuracy SVM_POLY : 79.31%
```

```
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.9529848171152518
AUC-ROC DT score: 0.7289222452265931
AUC-ROC ANN score: 0.9361858753163101
AUC-ROC NB score: 0.9224522659305268
AUC-ROC SVM_RBF score: 0.9108005521049
AUC-ROC SVM_LINEAR score: 0.8420519898780768
AUC-ROC SVM_POLY score: 0.9001150218541523
```

```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
# print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.6958333333333333
Mean score SVM LINEAR: 0.6291666666666667
Mean score SVM POLY: 0.655
Mean score KNN: 0.7575000000000001
Mean score ANN: 0.7575
Mean score NB : 0.6824999999999999
```

- modèle 9 (question 9) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
```

```
Accuracy KNN : 83.33%
Accuracy DT : 55.56%
Accuracy ANN : 83.33%
Accuracy NB : 83.33%
Accuracy SVM_RBF : 77.78%
Accuracy SVM_LINEAR : 38.89%
Accuracy SVM_POLY : 50.00%
```

```
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.9807900432900433
AUC-ROC DT score: 0.6720779220779219
AUC-ROC ANN score: 0.9761904761904762
AUC-ROC NB score: 0.9512987012987013
AUC-ROC SVM_RBF score: 0.9512987012987013
AUC-ROC SVM_LINEAR score: 0.9307359307359307
AUC-ROC SVM_POLY score: 0.7521645021645021
```



```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
print("Mean score DT: ", cross_val_score(DT, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.8733333333333333
Mean score SVM LINEAR: 0.5685714285714286
Mean score SVM POLY: 0.5961904761904762
Mean score ANN: 0.7761904761904762
Mean score NB : 0.8733333333333334
Mean score KNN: 0.7761904761904762
```

## • modèle 10 (question 10) :

```
## -----Classification Accuracy-----:
print("Accuracy KNN : {:.2f}%".format(accuracy_score(Y_test, y_pred_KNN)*100))
print("Accuracy NB : {:.2f}%".format(accuracy_score(Y_test, y_pred_NB)*100))
print("Accuracy SVM_RBF : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_RBF)*100))
print("Accuracy SVM_LINEAR : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_LINEAR)*100))
print("Accuracy SVM_POLY : {:.2f}%".format(accuracy_score(Y_test, y_pred_SVM_POLY)*100))
print("Accuracy ANN : {:.2f}%".format(accuracy_score(Y_test, y_pred_ANN)*100))
print("Accuracy DT : {:.2f}%".format(accuracy_score(Y_test, y_pred_DT)*100))
```

```
Accuracy KNN : 73.33%
Accuracy NB : 53.33%
Accuracy SVM_RBF : 66.67%
Accuracy SVM_LINEAR : 60.00%
Accuracy SVM_POLY : 60.00%
Accuracy ANN : 73.33%
```

```
print("Mean score RBF: ", cross_val_score(SVM_RBF, X_train, Y_train, cv=5).mean())
print("Mean score SVM LINEAR: ", cross_val_score(SVM_LINEAR, X_train, Y_train, cv=5).mean())
print("Mean score SVM POLY: ", cross_val_score(SVM_POLY, X_train, Y_train, cv=5).mean())
print("Mean score ANN: ", cross_val_score(ANN, X_train, Y_train, cv=5).mean())
print("Mean score NB : ", cross_val_score(NB, X_train, Y_train, cv=5).mean())
print("Mean score KNN: ", cross_val_score(KNN, X_train, Y_train, cv=5).mean())
```

```
Mean score RBF: 0.6588235294117647
Mean score SVM LINEAR: 0.3647058823529412
Mean score SVM POLY: 0.6117647058823529
Mean score ANN: 0.6
Mean score NB : 0.611764705882353
Mean score KNN: 0.5529411764705883
```

```
y_scores_KNN = KNN.predict_proba(X_test)
y_scores_DT = DT.predict_proba(X_test)
y_scores_ANN = ANN.predict_proba(X_test)
y_scores_NB = NB.predict_proba(X_test)
y_scores_SVM_RBF = SVM_RBF.predict_proba(X_test)
y_scores_SVM_LINEAR = SVM_LINEAR.predict_proba(X_test)
y_scores_SVM_POLY = SVM_POLY.predict_proba(X_test)
# Affichage du score AUC-ROC
print('AUC-ROC KNN score:', roc_auc_score(Y_test, y_scores_KNN, multi_class='ovr'))
print('AUC-ROC DT score:', roc_auc_score(Y_test, y_scores_DT, multi_class='ovr'))
print('AUC-ROC ANN score:', roc_auc_score(Y_test, y_scores_ANN, multi_class='ovr'))
print('AUC-ROC NB score:', roc_auc_score(Y_test, y_scores_NB, multi_class='ovr'))
print('AUC-ROC SVM_RBF score:', roc_auc_score(Y_test, y_scores_SVM_RBF, multi_class='ovr'))
print('AUC-ROC SVM_LINEAR score:', roc_auc_score(Y_test, y_scores_SVM_LINEAR, multi_class='ovr'))
print('AUC-ROC SVM_POLY score:', roc_auc_score(Y_test, y_scores_SVM_POLY, multi_class='ovr'))
```

```
AUC-ROC KNN score: 0.8976301476301476
AUC-ROC DT score: 0.8573556073556073
AUC-ROC ANN score: 0.8448804282137615
AUC-ROC NB score: 0.7557411724078391
AUC-ROC SVM_RBF score: 0.940796857463524
AUC-ROC SVM_LINEAR score: 0.1962358629025296
AUC-ROC SVM_POLY score: 0.8294051627384961
```

## Resumé :











Modèle	Algorithme adéquat	Accuracy	Auc-Roc	Cross Validation
1	KNN	94%	0.98	79%
2	SVM_RBF	89%	0.99	92%
3	KNN	74	0.89	56%
4	Naive Bayes	82%	0.89	60%
5	ANN	80%	0.85	66%
6	KNN	90%	0.96	90%
7	ANN	93%	1	89%
8	SVM_RBF	89%	0.91	70%
9	Naive Bayes	83%	0.95	87%
10	KNN	74	0.89	56%

## Remarque :

Parfois, il y a des modèles où la valeur de validation croisée n'est pas élevée, mais où l'AUC-ROC et l'Accuracy sont élevées. Cela ne pose pas le problème d'Overfitting (surapprentissage), car ces modèles sont bien testés sur de nouvelles données.

## Enregistrement des modèles :

```
import pickle
model_path = 'C:/Users/HP/Desktop/save_models/model_question4.h5'
with open(model_path, 'wb') as file:
    pickle.dump((NB, model), file)
```

 model_question1.h5	16/05/2023 23:08	Fichier H5	524 Ko
 model_question2.h5	17/05/2023 14:43	Fichier H5	307 Ko
 model_question3.h5	17/05/2023 14:49	Fichier H5	419 Ko
 model_question4.h5	17/05/2023 14:49	Fichier H5	391 Ko
 model_question5.h5	17/05/2023 14:50	Fichier H5	651 Ko
 model_question6.h5	17/05/2023 14:50	Fichier H5	464 Ko
 model_question7.h5	17/05/2023 14:52	Fichier H5	306 Ko
 model_question8.h5	17/05/2023 14:53	Fichier H5	360 Ko
 model_question9.h5	17/05/2023 14:53	Fichier H5	422 Ko
 model_question10.h5	21/05/2023 17:38	Fichier H5	326 Ko

## Déploiement du modèle :

Dans ce projet, on a utilisé FastAPI pour développer le backend de notre application et Angular pour le frontend. Nous avons choisi FastAPI en raison de sa performance élevée et de sa prise en charge intégrée de GraphQL, qui nous a permis de mettre en place une API GraphQL pour la communication entre le frontend et le backend.

### ➤ FastAPI :

```
from fastapi import FastAPI
from strawberry.asgi import GraphQL
import strawberry
from fastapi.middleware.cors import CORSMiddleware
from typing import List
import models.ready_model as ready_model

app = FastAPI()
```

L'importation de dix modules :



- ▼ models
  - > \_\_pycache\_\_
  - ≡ model\_question1.h5
  - ≡ model\_question2.h5
  - ≡ model\_question3.h5
  - ≡ model\_question4.h5
  - ≡ model\_question5.h5
  - ≡ model\_question6.h5
  - ≡ model\_question7.h5
  - ≡ model\_question8.h5
  - ≡ model\_question9.h5
  - ≡ model\_question10.h5
  - 🔗 ready\_model.py

```

ready_model.py X
models > 🔗 ready_model.py > 📦 predict
1  import pickle
2  import numpy as np
3  from nltk.tokenize import word_tokenize
4  import nltk
5  nltk.download('punkt')
6
7  import warnings
8  warnings.filterwarnings('ignore')
9
10 def predict(input_ans , num_ques):
11     model_path = "model_question"+num_ques + ".h5"
12     with open(model_path, 'rb') as file:
13         model, model_word2vec = pickle.load(file)
14     input_ans = preprocces_input(input_ans, model_word2vec)
15     input_ans = input_ans.reshape(1, -1)
16     pred = model.predict(input_ans)
17     result = pred[0]
18     return result
19

```

```

def get_word_vector(tokens, model_word2vec):
    textvector = np.zeros((100,), dtype='float32')
    for token in tokens:
        try:
            textvector += model_word2vec.wv[token]
        except KeyError:
            continue
    return textvector

def preprocces_input(text, model_word2vec):
    text = text.lower()
    tokens = word_tokenize(text)
    textvector = get_word_vector(tokens, model_word2vec)
    return textvector

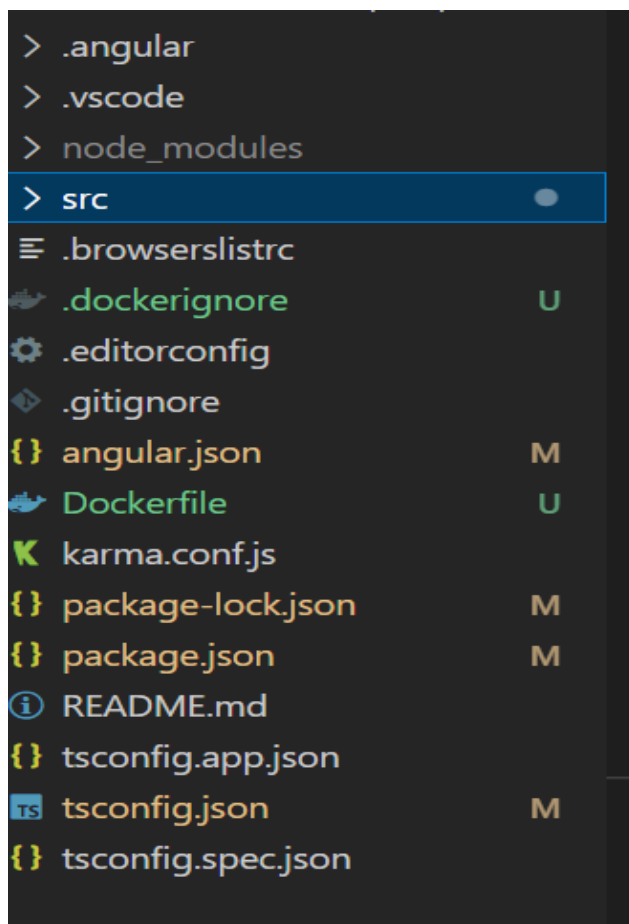
```

La prédiction du score de chaque réponse et le retourner au front :

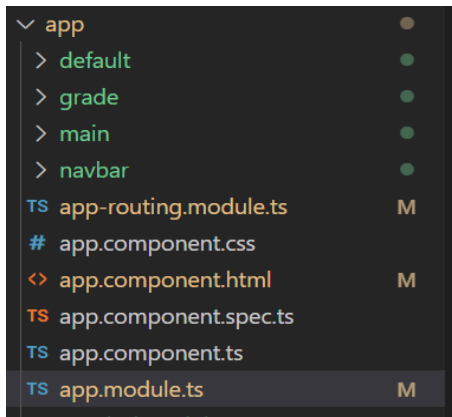
```
@strawberry.type
class Mutation:
    @strawberry.mutation
    def create_question(self, id: str, answer: str) -> List[Question]:
        if(id=="1"):
            questions.clear()
            score=ready_model.predict(answer, id)
            questions.append(Question(id=id, answer=answer, score=score))
        return questions
```

### ➤ Angular :

La structure du projet Angular :



Dans App on y crée quatre component : page principale, page des questions, page des réponses et le navbar :



## ➤ GraphQL :

Pour faciliter la communication entre le frontend et le backend, on a utilisé GraphQL.

### Intégration en fastApi (coté backend) :

On a utilisé Strawberry qui est une bibliothèque Python qui permet de créer des serveurs GraphQL de manière simple et intuitive. Elle offre une syntaxe déclarative pour définir des schémas GraphQL, des types de données, des requêtes et des résolveurs.

```
@strawberry.type
class Question:
    id: str
    answer: str
    score: int

questions: List[Question] = []

@strawberry.type
class Query:
    @strawberry.field
    def questions(self) -> List[Question]:
        return questions

@strawberry.type
class Mutation:
    @strawberry.mutation
    def create_question(self, id: str, answer: str) -> List[Question]:
        if(id=="1"):
            questions.clear()
            score=ready_model.predict(answer, id)
            questions.append(Question(id=id, answer=answer, score=score))
        return questions
```

```

schema = strawberry.Schema(query=Query, mutation=Mutation)

graphql_app = GraphQL(schema)
app.add_route("/graphql", graphql_app)
app.add_websocket_route("/graphql", graphql_app)

```

## Intégration en Angular (Côté frontend) :

### graphql.module :

Apollo facilite l'exécution de requêtes GraphQL, la gestion du cache, la souscription aux mises à jour en temps réel et bien plus encore. Il fournit également des outils et des composants pour faciliter l'intégration d'Apollo dans votre application Angular, ce qui vous permet de développer rapidement des fonctionnalités basées sur GraphQL.

```

import { APOLLO_OPTIONS, ApolloModule } from 'apollo-angular';
import { HttpLink } from 'apollo-angular/http';
import { NgModule } from '@angular/core';
import { ApolloClientOptions, InMemoryCache } from '@apollo/client/core';

const uri = 'http://127.0.0.1:8000/graphql'; // <-- add the URL of the GraphQL server here
export function createApollo(httpLink: HttpLink): ApolloClientOptions<any> {
  return {
    link: httpLink.create({ uri }),
    cache: new InMemoryCache(),
  };
}

@NgModule({
  exports: [ApolloModule],
  providers: [
    {
      provide: APOLLO_OPTIONS,
      useFactory: createApollo,
      deps: [HttpLink],
    },
  ],
})
export class GraphQLModule {}

```

### Graphql.operations :

Les constantes GET\_ANSWERS et CREATE\_QUESTION contiennent les requêtes pour récupérer des réponses de questions existantes et créer une nouvelle question, respectivement.

```
import { gql } from 'apollo-angular'

const GET_ANSWERS=gql`

query {
  questions {
    id
    answer
    score
  }
}

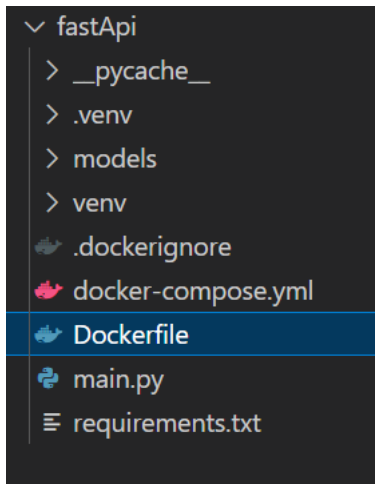
`

const CREATE_QUESTION = gql`
mutation($id: String!, $answer: String!) {
  createQuestion(id: $id, answer: $answer) {
    id
    answer
    score
  }
}
`

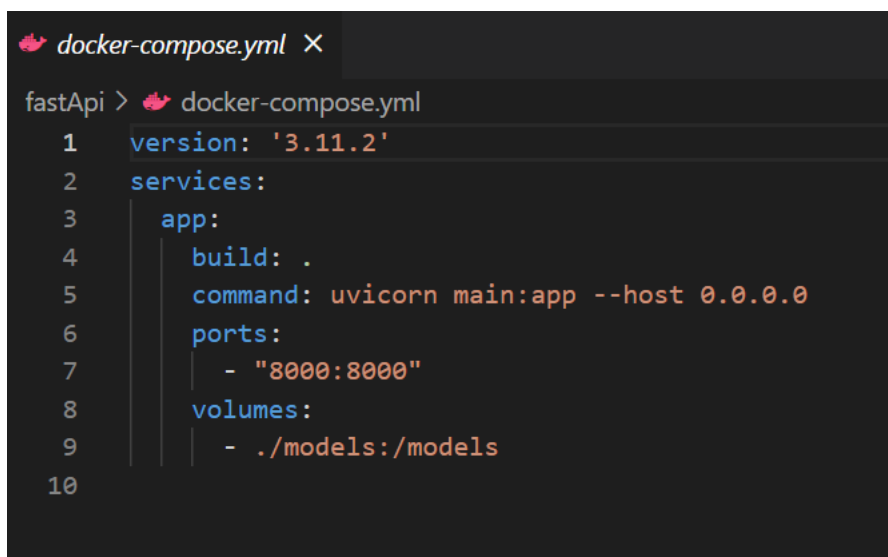
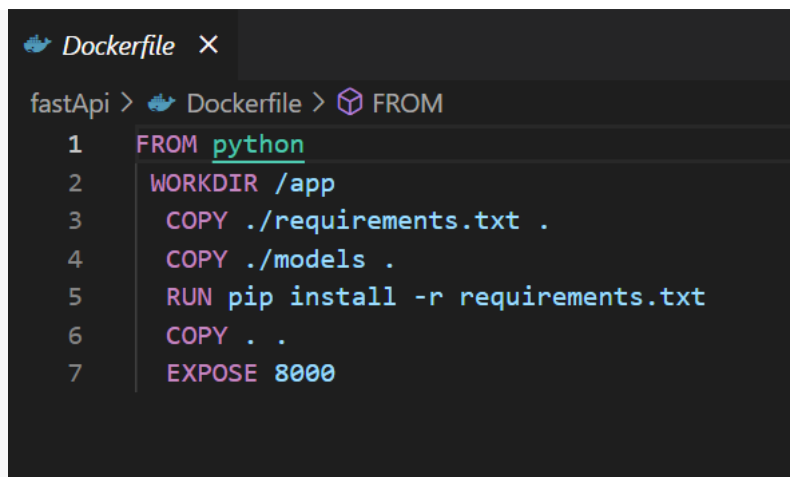
export { GET_ANSWERS, CREATE_QUESTION}
```

## Mise en conteneur avec Docker

### ➤ Containerisation du backend avec Docker :













## a) Construction de l'image Docker :



## b) Création du conteneur Docker :

```
(.venv) PS C:\Users\DELL LATITUDE\Desktop\docker\back\fastApi> docker-compose up
[+] Building 157.8s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 34B
=> [internal] load metadata for docker.io/library/python:latest
=> [1/6] FROM docker.io/library/python@sha256:b9683fa80e22970150741c974f45bf1d25856bd76443ea561df4e6fc00c2bc17
=> [internal] load build context
=> => transferring context: 1.31MB
=> CACHED [2/6] WORKDIR /app
=> CACHED [3/6] COPY ./requirements.txt .
=> [4/6] COPY ./models .
=> [5/6] RUN pip install -r requirements.txt
=> [6/6] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:c8a80ed85c5d56746ab732ca99f4d28ed08b5bd76ad806c2a77b6e161d515ce7
=> => naming to docker.io/library/fastapi-app
```

```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 2/2
 - Network fastapi_default Created
 - Container fastapi-app-1 Created
Attaching to fastapi-app-1
fastapi-app-1 | [nltk_data] Downloading package punkt to /root/nltk_data...
fastapi-app-1 | [nltk_data] Unzipping tokenizers/punkt.zip.
fastapi-app-1 | INFO: Started server process [1]
fastapi-app-1 | INFO: Waiting for application startup.
fastapi-app-1 | INFO: Application startup complete.
fastapi-app-1 | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

<input type="checkbox"/>	 fastapi	-	Running (1/1)	  
<input type="checkbox"/>	 app-1 141b684142ab 	<a href="#">fastapi-app:latest</a>	Running <a href="#">8000:8000</a> 	3 minutes ago   

## ➤ Containerisation du frontend avec Docker :

















### a) Construction de l'image Docker :

```
Dockerfile > ...
1  # Build stage
2  FROM node:18.13.0-alpine as build
3  RUN mkdir -p /app
4
5  WORKDIR /app
6
7  COPY package.json /app/
8  RUN npm install
9
10 COPY . /app/
11 RUN npm run build --prod
12
13 # Prod stage
14 FROM nginx:alpine
15 COPY --from=build /app/dist/test2 /usr/share/nginx/html
```

## b) Création du conteneur Docker :

```
PS C:\Users\DELL LATITUDE\Desktop\docker\front\test2> docker build --tag angularapp .
[+] Building 53.7s (15/15) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 32B                                                0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 348                                                    0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine                  2.4s
=> [internal] load metadata for docker.io/library/node:18.13.0-alpine            2.4s
=> CACHED [stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:02ffd439b71d9ea9408e449b568f65c0bbbb94bebd8750f1d80 0.0s
=> [internal] load build context                                                  0.3s
=> => transferring context: 99.91kB                                               0.2s
=> [build 1/7] FROM docker.io/library/node:18.13.0-alpine@sha256:fda98168118e5a8f4269efca4101ee51dd5c75c0fe56d8eb6fad8 0.0s
=> CACHED [build 2/7] RUN mkdir -p /app                                           0.0s
=> CACHED [build 3/7] WORKDIR /app                                                0.0s
=> CACHED [build 4/7] COPY package.json /app/                                    0.0s
=> CACHED [build 5/7] RUN npm install                                             0.0s
=> [build 6/7] COPY . /app/                                                       2.1s
=> [build 7/7] RUN npm run build --prod                                          48.4s
=> [stage-1 2/2] COPY --from=build /app/dist/test2 /usr/share/nginx/html         0.1s
=> exporting to image                                                            0.1s
=> => exporting layers                                                            0.1s
=> => writing image sha256:51d92733c135e6dc50b41bd4dbcd64c2cc7d46c4c4913ba511d07b5eda776002 0.0s
=> => naming to docker.io/library/angularapp                                     0.0s
```

```
PS C:\Users\DELL LATITUDE\Desktop\docker\front\test2> docker image ls
REPOSITORY      TAG         IMAGE ID      CREATED        SIZE
angularapp      latest      51d92733c135  About a minute ago  41.5MB
fastapi-app     latest      52ab36cf7c60  10 minutes ago  1.89GB
PS C:\Users\DELL LATITUDE\Desktop\docker\front\test2> docker run -d -p 4200:80 --name angular angularapp
78a58177cc8d01c054530d0de89654e0c4f5720c6380b4350f4482235ccb4f37
PS C:\Users\DELL LATITUDE\Desktop\docker\front\test2>
```

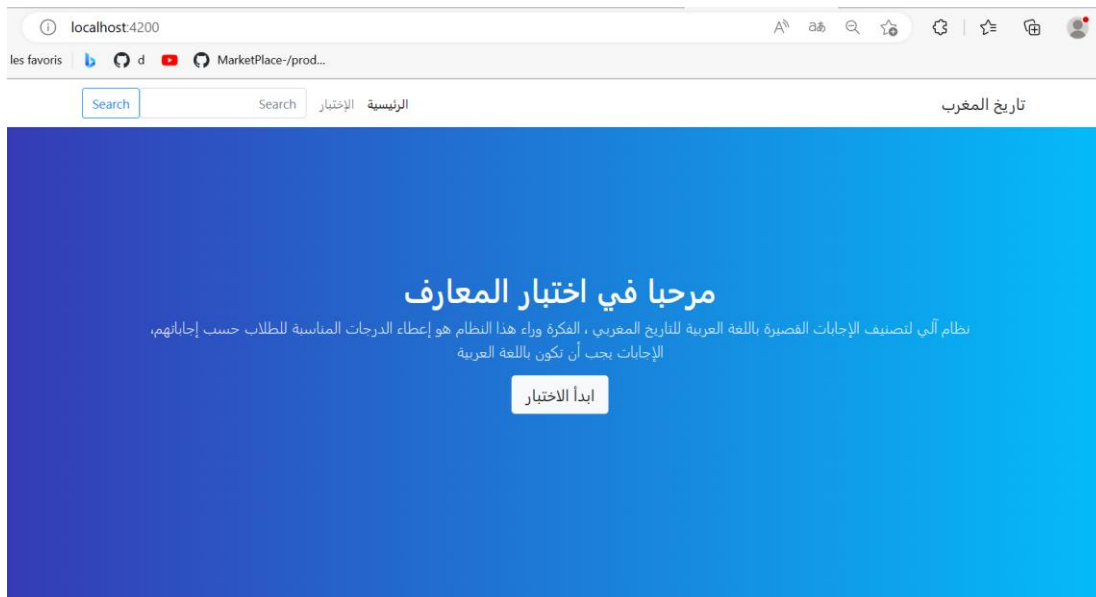
<input type="checkbox"/>		<b>angular</b> 78a58177cc8d 	<a href="#">angularapp:latest</a>	Running	<a href="#">4200:80</a> 	30 seconds ago 		
<input checked="" type="checkbox"/>		<b>fastapi</b>	-	Running (1/1)				
<input type="checkbox"/>		<b>app-1</b> 141b684142ab 	<a href="#">fastapi-app:latest</a>	Running	<a href="#">8000:8000</a> 	12 minutes ago 		

Maintenant on va tester le model après le lancement des deux applications

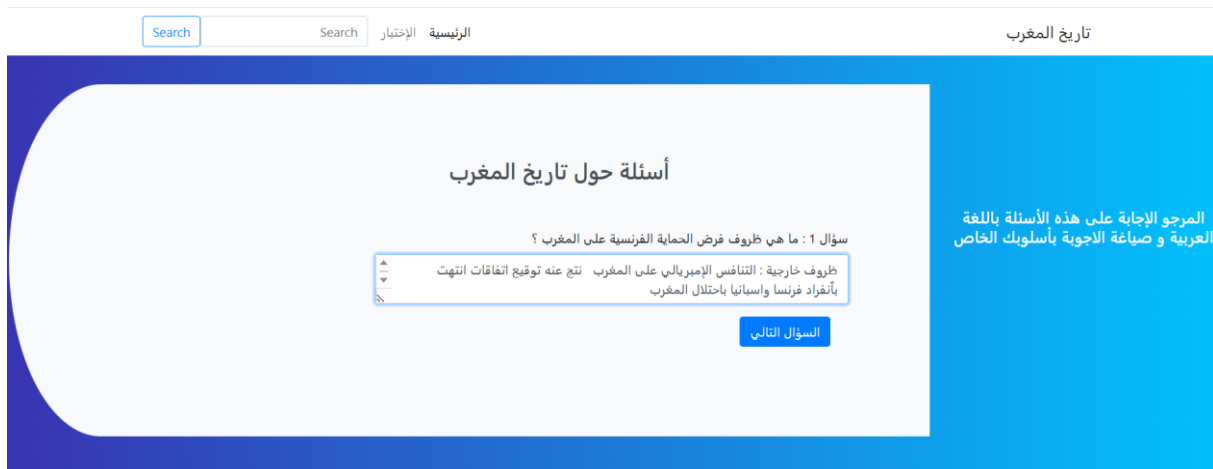


## Application

La page principale du site :



La page pour répondre aux 10 questions :



## أسئلة حول تاريخ المغرب

سؤال 4 : ما مظاهر الاستغلال الاستعماري في المجال الاقتصادي للمغرب ؟

مظاهر الاستغلال الاستعماري في المجال الاقتصادي للمغرب تتمثل في الاستيلاء على الموارد الطبيعية مثل الفحم والفوسفات وتصديرها بأسعار منخفضة لصالح الاستعمار الفرنسي، وتعرض الصناعة المحلية للمنافسة الشديدة مما أدى إلى تراجعها وتوقف العديد من المصانع.

السؤال التالي

## أسئلة حول تاريخ المغرب

سؤال 10 : ما نوعية الاصلاحات الاجتماعية التي باشرها المغرب خلال القرن 19 م ؟

قام المغرب بتحسين البنية التحتية للمدن لتحسين الحياة الاجتماعية

أحصل على التقييم

Après avoir répondu à toutes les questions on obtient le résultat total et le résultat pour chaque question :

(Réponses de personne 1)

## نهاية الإختبار

بداية إختبار جديد

## لقد حصلت على نتيجة 15/20

سؤال 1 : ما هي ظروف فرض الحماية الفرنسية على المغرب ؟

ظروف خارجية : التنافس الإمبريالي على المغرب نتج عنه توقيع اتفاقات انتهت بأنفراد فرنسا وإسبانيا باحتلال المغرب الاتفاق الفرنسي الإيطالي 1902 حول المغرب وليبيا الاتفاق الفرنسي الألماني 1911 حول المغرب والكونغو ظروف داخلية : ضعف السلطة المركزية بعد وفاة حاجب السلطان بإحماد 1900 وتوالي السلاطين في وقت قصير تزايد عدد ثورات والتمردات مثل تمرد الريسوني والجيلالي الزرعوني تزايد حجم القروض الأجنبية وإثقال كاهل الفلاحين بالضرائب انتشار الفقر والأمراض والأوبئة والبطالة والمجاعة

نتيجة = 2

سؤال 2 : ما هي مراحل الاحتلال العسكري للمغرب ؟

قبل 1912 تم احتلال الدار البيضاء ووحدة الرباط وفاس ما بين 1912 و1914 سيطرة القوات الفرنسية على وسط البلاد ما بين 1914 و1920 احتلال مناطق الأطلس المتوسط ما بين 1922 و1934 احتلال إسبانيا مناطق أقصى الشمال والجنوب

نتيجة = 2

سؤال 5 : ما هي انعكاسات الاستغلال الاستعماري على الاقتصاد و المجتمع المغربيين ؟

تقسيم المغرب في عهد الحماية إلى ثلاث مناطق :منطقة الاحتلال الفرنسي تشمل وسط المغرب منطقة الاحتلال الإسباني تشمل شمال المغرب وجنوب المغرب منطقة طنجة الدولية تزايد البنوك الأجنبية التي سهلت إنجاز البنيات التحتية وتحقيق أرباح انتزاع الأراضي من أصحابها واستغلالها من طرف المستعمرين الإستغلال المكثف للثروات المعدنية والطاقة واحتكار جميع وسائل الإنتاج

نتيجة = 1

سؤال 6 : ما هي ظروف ظهور الحركة الوطنية ؟

على مستوى الإقتصادي : تراجع الإنتاجين الفلاحي والصناعي المغربي توسع المساحات المزروعة من طرف المستعمرين غزو الأسواق المغربية بالبضائع والمنتجات الأجنبية عجز الميزان التجاري على مستوى الاجتماعي: تضرر الفلاحين المغاربة وجعلهم أجراء لدى المستعمرين تدهور وضعية الصناع والحرفيين وتحولهم لأجراء انتشار البطالة والمجاعة

نتيجة = 1

سؤال 7 : ما هي ظروف تقديم المغرب لوثيقة الاستقلال ؟

عد استجابة السلطان لرغبة الإقامة العامة وأقدمت على نفيه من 20 غشتى 1953م إلى جزيرة كوريسكا ثم إلى مدغشقر ونصبت محمد بن عرفة ملكا صوريا على البلاد

نتيجة = 0

سؤال 8 : ما علاقة ثورة الملك و الشعب باستقلال المغرب ؟

ثورة الملك والشعب هي حركة تحررية اندلعت في المغرب ضد الاحتلال الفرنسي، وقادها الملك محمد الخامس والشعب المغربي، وأسفرت في النهاية عن استقلال المغرب.

نتيجة = 2

(Réponses de personne 2)

## لقد حصلت على نتيجة 9/20

سؤال 1 : ما هي ظروف فرض الحماية الفرنسية على المغرب ؟

كانت الحماية الفرنسية تهدف إلى إخضاع المغرب للسيطرة الفرنسية وإيقاف النضال الوطني.

نتيجة = 0

سؤال 2 : ما هي مراحل الاحتلال العسكري للمغرب ؟

سيطرت القوات الفرنسية على وسط البلاد (فاس ومراكش وأحوازهما...). مرحلة ما بين 1914/1920م: احتلال مناطق الأطللس المتوسط.

نتيجة = 1

سؤال 3 : اذكر اسماء اهم المقاومين المغاربة و بين دورهم في مواجهة الاحتلال ؟

موحاً أوجمو الزياتي: من أبرز زعماء المقاومة المسلحة بالأطلس المتوسط، ولد حوالي 1877.

نتيجة = 0

## نهاية الإختبار

بداية إختبار جديد

## الاحتلال العسكري بالقوة

نتيجة = 1

سؤال 7 : ما هي ظروف تقديم المغرب لوثيقة الاستقلال ؟

بعد أن قدم أعضاء كتلة العمل الوطني وثيقة مطالب الشعب المغربي في 1 دجنبر 1934، تطورت هذه المطالب بعد تغير المشهد العام في المغرب، حيث تعرض بعض قادة الحركة الوطنية في نهاية الثلاثينات

نتيجة = 1

سؤال 8 : ما علاقة ثورة الملك و الشعب باستقلال المغرب ؟

المقاومة التي قادها الملك محمد الخامس والشعب المغربي ساهمت في نهاية المطاف في استعادة استقلال المغرب.

نتيجة = 2

سؤال 9 : ما هي الضغوط العسكرية التي استعملتها الدول الامبريالية للتغلغل الاوروبي في المغرب خلال القرن 19 م ؟

كان المغرب محط أطماع الدول الامبريالية ما جعله عرضة لضغوط استعمارية قسمت ما بين الضغوط العسكرية الفرنسية المتمثلة في معركة ايسلي سنة 1844 م التي انتهت بانهزام المغرب وتوقيع معاهدة للامغنية سنة 1845 م

نتيجة = 2

سؤال 10 : ما نوعية الاصلاحات الاجتماعية التي باشرها المغرب خلال القرن 19 م ؟

قام المغرب بمدارس جديدة لتعليم الفتيات

نتيجة = 0