Master 2 MoSEF, 2023

# Projet Machine Learning Avancé

ABABII Anisoara, KOFFI Eunice, DIAKITE Gaoussou, BOUKIR Amine

## *Présentation de la problématique*

Ce projet consiste en la prédiction de commentaires sur un film. Nous cherchons à prédire si un commentaire est positif ou négatif. Disposant de données textuelles, nous allons mettre en place une analyse complète sur données textuelles depuis le chargement de la base de données jusqu'à l'évaluation des modèles choisis.

Ce projet fait partie intégrante de notre cours de Machine Learning avancé au cours duquel nous avons appris à travailler les données textuelles pour leur appliquer des modèles allant des plus simples aux plus complexes. Cette pratique nous a démontré la nécessité de travailler avec un environnement nous permettant d'avoir aisément à disposition des ressources nécessaires aux traitements aussi complexes et gourmands de RAM qu'ils puissent être.

Le projet sera donc effectué sous Google Colab puis, importer en local, pour être peaufiner afin de rendre un travail de qualité.

Ce projet étant réalisé dans un groupe de 3 personnes, nous avons pu mettre ensemble nos connaissances et notre sens critique pour analyser au mieux les données que nous avions.

## *Présentation des données*

Le dataset "imdb_Movie_Expressed_Sentiment_2" est un sous-ensemble de données extraites de la base de données IMDB (Internet Movie Database). Il contient des critiques de films extraites de la base de données IMDB, avec des annotations de sentiment associées à chaque critique.

Les annotations de sentiment sont binaires (positive ou négative) et sont basées sur les notes données par les utilisateurs de IMDB aux films. Ce dataset peut être utilisé pour l'entraînement et l'évaluation des modèles d'analyse de sentiment en langage naturel.

Plus précisément, il peut être utilisé pour entraîner des modèles de classification binaire pour prédire si une critique de film est positive ou négative. Les modèles d'analyse de sentiment peuvent être utilisés pour de nombreuses applications telles que l'aide à la décision dans les entreprises, la compréhension des opinions des clients, et la détection d'émotions dans les médias sociaux.

Nous disposons donc de deux datasets : train et test. Nous avons décidé dans la suite de notre analyse de les traiter séparément pour éviter de les concaténer puis les splitter à nouveau.

## *Sélection de variables*

Nous avons sélectionné que deux variables qui nous servirons dans la suite de notre analyse. Ce sont les variables inputs_pretokenized pour le corpus et targets_pretokenized pour la target à prédire.
Les base de données ne contient aucune valeur manquante. Elles sont équilibrées. Nous avons les mê mes proportions que ce soit dans le train/test ou positif/négatif.

```
*********Dimensions des bases de données********
--------------  Train ------------------------
(25000, 2)
--------------  Test -------------------------
(25000, 2)
```

```
---------------  Target ------------------------
---------------  Train -------------------------
Train :  positive    12500
 negative    12500
Name: targets_pretokenized, dtype: int64
---------------  Test --------------------------
Test :  positive    12500
 negative    12500
Name: targets_pretokenized, dtype: int64
```

## *Nettoyage des données*

La première remarque que nous avons faite en analysant notre jeu de données et de constater la prés ence d'une phrase au début de chaque commentaire. Etant au début de chaque commentaire, nous l'a vons supprimé pour éviter d'alourdir la base de données.

Dans la suite de notre traitement, nous avons également supprimé de notre corpus, tous les liens UR Ls, HTML et les mots vides (stopwords).
Cela nous permet de réduire à minima le bruit que contient notre corpus et aussi de permettre une m eilleure comparaison et prise en compte des mots en les mettant tous à la même orthographe (tout en minuscule).

## *Quelques statistiques descriptives*

| | | targets_pretokenized | negative | positive |
|---|---|---|---|---|
| nombre_phrases | count | 12500.000000 | 12500.000000 |
| | mean | 9.862560 | 9.205760 |
| | std | 5.220170 | 5.120852 |
| | min | 1.000000 | 1.000000 |
| | 25% | 6.000000 | 6.000000 |
| | 50% | 9.000000 | 8.000000 |
| | 75% | 13.000000 | 12.000000 |
| | max | 51.000000 | 142.000000 |

| | | | |
|---|---|---|---|
| nombre_mots_moyen_par_phrase | count | 12500.000000 | 12500.000000 |
| | mean | 29.225371 | 31.120735 |
| | std | 24.102915 | 25.437111 |
| | min | 5.625000 | 2.098592 |
| | 25% | 18.833333 | 19.400000 |
| | 50% | 24.333333 | 25.600000 |
| | 75% | 32.200000 | 34.200000 |
| | max | 542.000000 | 454.000000 |

| | | | |
|---|---|---|---|
| nombre_total_mots | count | 12500.000000 | 12500.000000 |
| | mean | 244.555040 | 242.081760 |
| | std | 116.856489 | 119.790987 |
| | min | 11.000000 | 14.000000 |
| | 25% | 154.000000 | 149.000000 |
| | 50% | 211.000000 | 208.500000 |
| | 75% | 338.000000 | 352.000000 |
| | max | 596.000000 | 580.000000 |

| | | | |
|---|---|---|---|
| nombre_mots_uniques | count | 12500.000000 | 12500.000000 |
| | mean | 140.728720 | 138.467280 |
| | std | 54.626961 | 56.736703 |
| | min | 11.000000 | 12.000000 |
| | 25% | 99.000000 | 95.000000 |
| | 50% | 128.000000 | 126.000000 |
| | 75% | 185.000000 | 189.000000 |
| | max | 266.000000 | 263.000000 |

Que nous montrent nos données ? Qu'est-ce qui différencie les commentaires positifs des commentai res négatifs ? Pouvons-nous noter cette différence immédiatement ?

Ce sont à ces questions que nous essaieront de répondre en analysant au plus près nos données.

Nous commençons par les plus basiques en comptant le nombre de phrases, de mots que contiennen t les commentaires.

La statistique descriptive nous montre que la proportion est presque la même pour les phrases négati ves et positives. Cela rend plus difficile la distinction entre les commentaires. En moyenne, nous avon s remarqué un nombre similaire de commentaires classés comme positifs et négatifs (environ 9 milli ons d'avis).

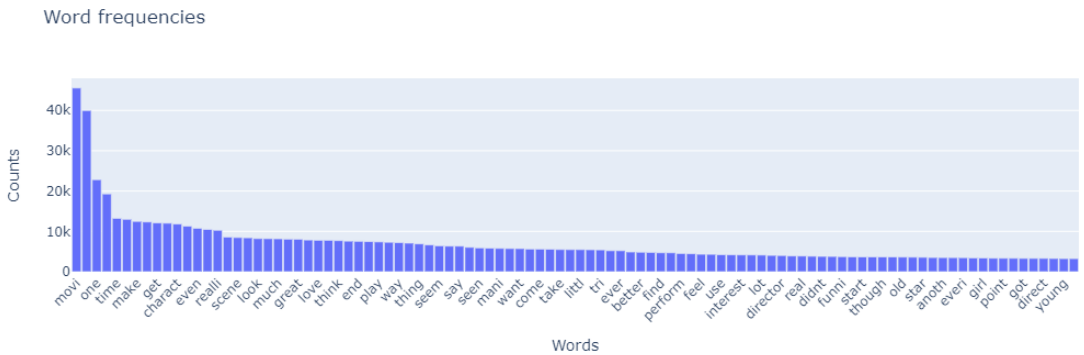
La description statistique fournie présente des mesures de tendance centrale et de dispersion pour de ux ensembles de données : les phrases négatives et les phrases positives. Les données comprennent l e nombre de phrases, le nombre moyen de mots par phrase, le nombre total de mots et le nombre de mots uniques.

En ce qui concerne le nombre de phrases, il y en a 12 500 pour chaque ensemble, avec une moyenne de 9,86 pour les phrases positives et 9,21 pour les phrases négatives. La dispersion des données est a ssez grande, avec une déviation standard de 5,22 pour les phrases positives et 5,12 pour les phrases n égatives. La valeur minimale pour les deux ensembles est de 1, tandis que la valeur maximale est de 5 1 pour les phrases positives et de 142 pour les phrases négatives.

En ce qui concerne le nombre moyen de mots par phrase, il y a une moyenne de 29,23 mots pour les phrases négatives et 31,12 mots pour les phrases positives. La déviation standard est également assez importante, avec 24,10 pour les phrases négatives et 25,44 pour les phrases positives. Les valeurs mi nimales et maximales pour les deux ensembles sont également très différentes, avec une valeur mini male de 2,10 pour les phrases négatives et 5,63 pour les phrases positives, tandis que la valeur maxim ale est de 542 pour les phrases négatives et de 454 pour les phrases positives.

En ce qui concerne le nombre total de mots, la moyenne est similaire pour les deux ensembles, avec 2 44,56 mots pour les phrases négatives et 242,08 mots pour les phrases positives. La déviation standa rd est également similaire, avec 116,86 pour les phrases négatives et 119,79 pour les phrases positives . Les valeurs minimales et maximales pour les deux ensembles sont également très différentes, avec u ne valeur minimale de 11 pour les phrases négatives et de 14 pour les phrases positives, tandis que la valeur maximale est de 596 pour les phrases négatives et de 580 pour les phrases positives.

Enfin, en ce qui concerne le nombre de mots uniques, la moyenne est également similaire pour les de ux catégories (positive/negative), avec 140,73 mots uniques pour les phrases négatives et 138,47 mot s uniques pour les phrases positives. La déviation standard est toutefois plus grande pour les phrases négatives, avec 56,74, contre 54,63 pour les phrases positives. Les valeurs minimales et maximales p our les deux ensembles sont également très différentes, avec une valeur minimale de 11 pour les phra ses négatives et de 12 pour les phrases positives, tandis que la valeur maximale est de 266 pour les ph rases négatives et de 263 pour les phrases positives.




Nous constatons que les mots les plus fréquents font partir du champ lexical du cinéma. Ce qui nous semble tout à fait normal. Cependant, jusqu'ici, nous n'avons noté aucun écart considérable qui nous permet de distinguer assez facilement les commentaires positifs des commentaires négatifs.

Nous décidons donc, d'approfondir nos recherches en construisant des nuages de mots différents sur les deux sentiments. Voici les résultats que nous obtenons :


En fond noir, nous avons les nuages de mots qui ont été faits à partir des commentaires complets et en blancs les nuages de mots faits à partir des adjectifs uniquement.
Dans les deux commentaires, nous constatons que nous avons quasiment les mêmes adjectifs avec une connotation positive.
Nous constatons d'ores et déjà en quoi consiste la complexité de l'analyse de données textuelles. En effet, nous pouvons noter que compte tenu de nos traitements faits, nous avons supprimé les stopwords parmi lesquels « not », « less », …
Cette suppression vient donc un peu biaiser notre analyse dans le sens où nous perdons de l'information. Il conviendra dans la suite de notre analyse de tenir compte de ces informations afin d'appliquer à nos datasets, les transformations adéquates pour l'entrainement et l'évaluation de nos modèles.

## *Modélisation*

- **Choix des modèles**

Nous avons opté pour une approche progressive dans notre choix de modèles, en commençant par les plus simples pour ensuite passer aux plus complexes. Nous allons ainsi explorer différentes catégories de modèles, telles que les méthodes à noyaux, les méthodes ensemblistes et les réseaux de neurones.

Il convient également de souligner que chaque modèle choisi nécessite une transformation spécifique. En effet, chaque méthode a ses propres paramètres et exigences, ce qui rend nécessaire une adaptation et une personnalisation des données en fonction des modèles sélectionnés. Nous allons donc prendre en compte ces facteurs pour garantir des résultats fiables et pertinents dans notre analyse.

En utilisant cette approche graduelle, nous espérons découvrir le modèle le plus adapté à notre jeu de données et ainsi être en mesure de fournir des résultats précis et significatifs.

- **La métrique d'évaluation**

Dans notre processus d'évaluation des différents modèles, nous avons opté pour l'utilisation de l'accuracy en tant que métrique d'évaluation. Cette mesure est essentielle pour arbitrer entre les différents modèles et ainsi identifier celui qui offre les meilleures performances.

En effet, l'accuracy nous permet de mesurer la précision des prédictions du modèle en comparant les résultats obtenus avec les valeurs réelles de notre jeu de données. Nous pourrons ainsi évaluer la performance de chaque modèle et déterminer celui qui présente les résultats les plus satisfaisants en termes d'exactitude.

Cependant, il est important de noter que l'accuracy ne peut pas être la seule mesure d'évaluation dans certains cas. Dans des situations où les classes sont déséquilibrées, il peut être nécessaire d'utiliser d'autres métriques telles que la précision, le rappel ou la F1-score pour avoir une évaluation plus précise des performances du modèle.

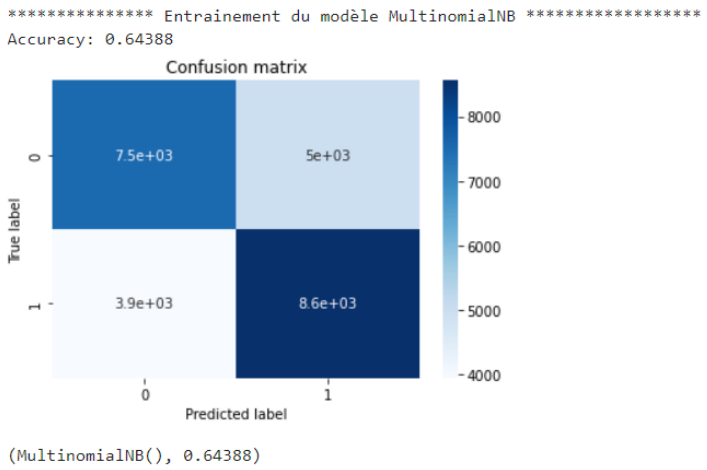
- **Naïve Bayes Multinomial**

Nous utilisons ce modèle en Benchmark pour tous les modèles que nous utiliserons.

Le modèle Naïve Bayes Multinomial est un algorithme de classification de textes qui se base sur la probabilité d'occurrence de chaque mot dans chaque classe (ou catégorie). Ce modèle est basé sur le théorème de Bayes, qui est une formule mathématique utilisée pour calculer la probabilité d'un événement en fonction des probabilités des événements antérieurs connus.

Dans le contexte de la classification de textes, le modèle Naïve Bayes Multinomial utilise le nombre de fois qu'un mot apparaît dans un document (fréquence du terme) pour estimer la probabilité d'appartenance de ce document à une certaine classe (par exemple, un document est-il plus susceptible d'appartenir à la classe "sport" ou à la classe "politique" ?). Le modèle suppose que la fréquence des termes suit une distribution multinomiale (d'où son nom) dans chaque classe.

Le modèle Naïve Bayes Multinomial est appelé "Naïve" car il suppose que la présence de chaque mot dans un document est indépendante des autres mots, ce qui est rarement vrai en pratique. Malgré cette limitation, le modèle Naïve Bayes Multinomial est très efficace pour la classification de textes en pratique, notamment grâce à sa simplicité, sa rapidité d'exécution et sa capacité à gérer de grandes quantités de données textuelles.




- **SVM.SVC**

Le modèle SVM.SVC est un modèle à noyau. Nous l'utilisons en première position. Nous choisissons pour ce modèle de laisser les paramètres par défaut. Ce modèle est entrainé en appliquant à notre features l'algorithme itf-idf comme traitement.

Un SVM.SVC (Support Vector Machine) est un algorithme d'apprentissage supervisé utilisé pour la classification binaire et multi-classe. Lorsqu'il est appliqué à des données textuelles pré-traitées à l'aide de TF-IDF, il apprend à classer les documents en fonction de leur similarité à des documents appartenant à différentes catégories.

Pour ce faire, le SVM.SVC utilise les vecteurs de caractéristiques représentant chaque document, où chaque caractéristique est un mot pondéré par sa fréquence inverse de document (TF-IDF). Le SVM.SVC cherche ensuite à séparer les vecteurs de caractéristiques de chaque catégorie en construisant un hyperplan qui maximise la marge entre les vecteurs de caractéristiques les plus proches de chaque catégorie.

Le SVM.SVC peut être utilisé pour la classification binaire (par exemple, spam ou non-spam) ou multi-classe (par exemple, classification de sujets de courrier électronique en plusieurs catégories). Il est largement utilisé pour la classification de données textuelles en raison de sa capacité à gérer des données avec de nombreuses caractéristiques (mots) et sa capacité à généraliser sur des données non vues.

Cependant, il est important de noter que le SVM.SVC ne prend pas en compte l'ordre des mots ou leur contexte, et qu'il peut ne pas être approprié pour des tâches de classification plus complexes où la sémantique et le contexte sont importants. De plus, il peut nécessiter un réglage fin des hyperparamètres pour obtenir les meilleurs résultats sur un ensemble de données spécifique.

Nous obtenons un accuracy 0.87 et la matrice de confusion est meilleure que la précédente.

- **LSTM**

Ce code Python définit et entraîne un modèle de réseau de neurones pour la classification de textes en utilisant l'architecture Embedding-LSTM. Pour ce modèle on utilise des techniques de prétraitement de texte telles que la tokenization et le padding, et utilise une représentation vectorielle de mots pré-entraînée à partir du modèle Word2Vec de Gensim pour initialiser les poids de la couche Embedding.

Les étapes d'entrainement du modèle sont les suivantes (très détaillées) :

Application d'un algorithme d'apprentissage profond Word2Vec qui sert dans ce cas à apprendre des représentations vectorielles de mots à partir de grands ensembles de données textuelles. Dans ce code, la fonction Word2Vec est utilisée pour entraîner un modèle Word2Vec à partir des données de texte pré-traitées (train_df2_select['preprocessed']). Le modèle apprend des vecteurs de mots de dimension 50 (vector_size=50) en utilisant une fenêtre de contexte de taille 5 (window=5) et un nombre minimum d'occurrences de 1 (min_count=1). Plus la taille de la dimension que nous choisissons est grande, plus le modèle peut capturer de nuances dans la signification des mots, mais cela peut également augmenter la complexité du modèle et sa durée d'entraînement. Par rapport à la taille de la fenêtre de contexte, elle est utilisée par le modèle avec le but d'apprendre les relations entre les mots. Le modèle considère le mot cible ainsi que les mots environnants dans la fenêtre de contexte spécifiée lors de l'entraînement. Ensuite, le paramètre min_count=1 indique le nombre minimum d'occurrences que doit avoir un mot dans le corpus pour être inclus dans le modèle Word2Vec. Les mots qui n'apparaissent pas assez souvent dans le corpus peuvent ne pas avoir suffisamment de contexte pour que le modèle puisse apprendre une représentation sémantique appropriée.

La classe Tokenizer de Keras est utilisée pour convertir les données textuelles en une séquence de nombres entiers. Les méthodes fit_on_texts et texts_to_sequences sont utilisées pour adapter le tokenizer aux données d'entraînement et pour convertir les données d'entraînement et de test en séquences d'entiers.

Ce que nous avons remarqué aussi c'est le fait que les séquences d'entiers ont des longueurs différentes, il faut donc les mettre à la même taille pour qu'elles puissent être entrées dans le modèle de manière cohérente. La fonction pad_sequences de Keras est utilisée pour ajouter des zéros à la fin des séquences plus courtes que la longueur maximale (max_length=50). Ensuite, la couche Embedding de Keras est utilisée pour créer une représentation vectorielle dense de chaque mot de la séquence d'entiers. La méthode zeros de NumPy est utilisée pour initialiser la matrice d'embedding (embedding_matrix) avec des zéros, puis les vecteurs de mots pré-entraînés à partir du modèle Word2Vec sont insérés dans la matrice aux positions correspondantes aux index des mots dans le tokenizer.

Concernant le modele proprement, les couches LSTM de Keras sont utilisées pour traiter la séquence d'embeddings de mots en conservant la mémoire à long terme et en utilisant une porte d'oubli pour contrôler l'information qui est conservée et celle qui est oubliée. Le modèle utilise deux couches LSTM, avec des paramètres de dropout (dropout=0.2, recurrent_dropout=0.2) pour éviter le surapprentissage.

Les couches Dense de Keras sont utilisées pour la classification. La fonction d'activation relu est utilisée pour les couches cachées (Dense(64, activation='relu'), Dense(32, activation='relu'), Dense(16, activation='relu')), tandis que la fonction d'activation sigmoid est utilisée pour la couche de sortie (Dense(1, activation='sigmoid')). Les couches Dense de Keras sont utilisées pour créer des réseaux de neurones artificiels (RNA) denses et connectés. Ces couches sont particulièrement utiles pour la classification de nos données textuelles, car elles permettent de réduire la dimensionnalité des données d'entrée (les vecteurs de mots) et d'extraire les caractéristiques pertinentes pour la tâche de classification. L'architecture du modele est representé comme suivant :

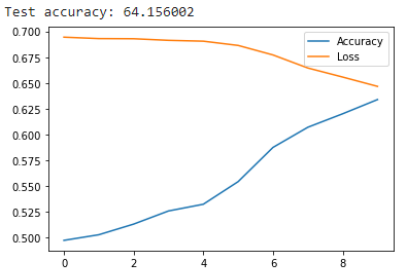
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 50, 50)            2840250

 lstm_2 (LSTM)               (None, 50, 128)           91648

 lstm_3 (LSTM)               (None, 64)                49408

 dense_4 (Dense)             (None, 64)                4160

 dropout_3 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 32)                2080

 dropout_4 (Dropout)         (None, 32)                0

 dense_6 (Dense)             (None, 16)                528

 dropout_5 (Dropout)         (None, 16)                0

 dense_7 (Dense)             (None, 1)                 17

=================================================================
Total params: 2,988,091
Trainable params: 147,841
Non-trainable params: 2,840,250
_________________________________________________________________
```

La fonction d'activation "relu" (rectified linear unit), fonction non linéaire qui est couramment utilisée pour les couches cachées des RNA, qui permettent ensuite d'introduire de la non-linéarité dans les réseaux de neurones et de faciliter l'apprentissage de représentations hiérarchiques de données complexes de NLP.

La fonction d'activation "sigmoid" , aussi fonction non linéaire qui est utilisée dan cette analyse de sentiment  pour la classification binaire de commentaires (positifs ou négatifs). Elle produit des valeurs en sortie comprises entre 0 et 1, qui peuvent être interprétées comme des probabilités. Dans le cas de ce modèle, la fonction sigmoid est utilisée pour prédire la probabilité d'appartenance à la classe positive (1) ou négative (0).

La méthode Dropout de Keras est utilisée pour régulariser le modèle et éviter le surapprentissage. Les paramètres dropout de 0,5 sont utilisés pour les couches Dense.

À la fin de l'entrainement du modele, on utilise la méthode compile de Keras pour compiler le modèle. L'optimiseur Adam est utilisé avec une fonction de perte binary_crossentropy pour la classification binaire. Le modele est ensuite entraîné sur les données d'entraînement et de validation (test). Pour ce modele, alors, on obtiens une accuracy = 64 % sur le test (resultat assez proche de Naive Bayesien Model).




## *Conclusions*

Notre projet d'analyse de sentiment a été une expérience fascinante et fructueuse, où nous avons exploré plusieurs modèles pour classifier les avis des utilisateurs. Nous avons commencé par le modèle Naïve Bayes Multinomial, un algorithme simple et rapide, mais qui n'a pas donné les meilleurs résultats.

C'est ensuite que nous avons utilisé le modèle SVM.SVC, un algorithme à noyau qui a su prendre en compte les spécificités des données textuelles pré-traitées en utilisant le TF-IDF. Avec ce modèle, nous avons obtenu le plus grand score **d'accuracy de 0,87**, et avons observé une nette amélioration de la matrice de confusion par rapport aux résultats précédents.

Nous avons également exploré le modèle LSTM, un réseau de neurones récurrents capable de prendre en compte l'ordre et le contexte des mots dans les phrases. Nous avons procédé à un fine-tuning minutieux des hyperparamètres, testant différentes architectures et paramètres pour obtenir le meilleur résultat possible. Bien que nous n'ayons pas obtenu de meilleurs résultats que le SVM.SVC, notre exploration des réseaux de neurones a été passionnante et nous a permis d'en apprendre davantage sur leur potentiel dans le traitement de données textuelles complexes. En conclusion, l'ajustement des hyperparamètres que nous avons fait c'est une étape essentielle pour obtenir des performances optimales d'un modèle de classification de données textuelles. La démarche implique une compréhension approfondie des caractéristiques du corpus de données et une exploration systématique des hyperparamètres pour trouver la combinaison optimale qui maximise les performances du modèle.

Il est essentiel de souligner que lors de notre étude, nous avons également exploré les avantages et les inconvénients d'un modèle de langage de pointe appelé qui est considéré comme une amélioration significative de BERT en termes de performances de traitement de langage naturel. Cependant, nous avons découvert que l'utilisation de RoBERTa était plus coûteuse en termes de temps et de ressources, notamment en termes de RAM, ce qui a posé des défis supplémentaires pour l'implémentation du modèle. Néanmoins, nous avons tiré des leçons importantes de cette expérience, notamment l'importance de l'évaluation de l'efficacité des modèles en fonction de leurs coûts en ressources et de leur faisabilité en termes de mise en œuvre.

En fin de compte, notre choix s'est porté sur le **modèle SVM.SVC**, qui s'est avéré être le meilleur pour classifier les avis des utilisateurs. Nous sommes convaincus que notre approche rigoureuse et méthodique a permis de mettre en évidence l'importance de la sélection du bon modèle et des hyperparamètres appropriés pour la classification de données textuelles. Nous espérons que notre projet inspirera d'autres chercheurs à explorer davantage les possibilités offertes par les modèles d'apprentissage automatique pour l'analyse de sentiment.

## *Contribution et acquis*

Au sein de notre groupe de travail, chacun de nous a contribué de manière significative à la réalisation de ce projet d'analyse de sentiment. Tout d'abord, nous avons appris à travailler en équipe et à communiquer efficacement tout au long du processus. Nous avons également acquis une compréhension approfondie des différents modèles de classification de textes que nous avons utilisés, en comprenant les avantages et les limites de chacun.

**Gaoussou** : En ce qui concerne ma contribution personnelle, j'ai joué un rôle clé dans l'élaboration de la méthodologie de travail et dans la mise en place des différents modèles de classification. J'ai également été impliqué dans l'analyse des résultats et dans l'interprétation des différentes métriques de performance. Enfin, j'ai contribué à la rédaction des rapports et des présentations finales, en synthétisant les résultats obtenus et en mettant en évidence les points forts et les limites de notre approche.

**Anisoara** : En tant que membre du groupe, ma contribution a été de participer activement à toutes les étapes du projet, de la collecte de données à l'évaluation des modèles. J'ai apporté mes connaissances en programmation et en statistiques pour aider à la mise en place des modèles et à l'interprétation des résultats. J'ai également travaillé en étroite collaboration avec les autres membres du groupe pour discuter des choix de modèles et des stratégies d'optimisation. Grâce à ce projet, j'ai appris à travailler en équipe sur un projet complexe et à appliquer mes connaissances en programmation et en statistiques à un problème réel. J'ai également acquis une meilleure compréhension des techniques d'analyse de sentiment et de leur application pratique. En somme, ce projet a été une expérience très enrichissante pour moi sur le plan personnel et professionnel.

**Eunice** : En tant que membre du groupe, ma contribution a été de mener une partie de l'analyse exploratoire des données, en utilisant des outils tels que la visualisation de données et l'extraction de mots-clés. J'ai également participé à la conception et à l'optimisation des modèles de classification, en utilisant des techniques telles que Naïve Bayes et SVM.SVC. Au cours de ce projet, j'ai appris l'importance de la préparation des données pour la réussite d'un projet d'analyse de sentiment, ainsi que l'importance de choisir la bonne métrique d'évaluation pour mesurer les performances des modèles. J'ai également acquis une meilleure compréhension des modèles de classification de textes, notamment de la manière dont ils fonctionnent et des avantages et des limitations de chaque méthode.

**Amine** : Au cours de ce projet de deeplearning sur l'analyse de sentiment, j'ai appris que la classification de texte peut être un processus complexe mais fascinant. En utilisant des techniques de traitement de langage naturel et de réseaux de neurones profonds, on a pu construire un modèle capable de prédire si un sentiment était positif ou négatif avec une précision raisonnable.

Dans l'ensemble, ce projet nous a permis d'acquérir de nouvelles compétences en matière de traitement des données textuelles et de classification de textes, ainsi que de renforcer notre capacité à travailler en équipe et à mener à bien des projets complexes. Nous sommes convaincus que les connaissances et les compétences acquises au cours de ce projet nous seront utiles dans nos futures carrières professionnelles.