MASTER
INFORMATIQUE

Traitement automatique du texte en IA
Classification de sommaire par genre

Auteurs:
Dylann Batisse
Margaux Schmied

Professeur:
Elena Cabrio

2021/2022

Table des matières

1 Introduction

Pour ce projet nous avions le choix du sujet, rapidement nous nous sommes orientés vers la classification de film par genre. Au début notre premier choix était de classer ces films grâce à leur scénario malheureusement nous n'avons pas réussi a trouvé une base de données avec assez de ressource variée. C'est pourquoi nous nous sommes finalement orientés vers une classification grâce au résumé.

Pour commencer nous avons dû sélectionner les bases de données adapté puis essayer plusieurs méthodes de classification.

2 Data

2.1 Base de donnée de résumé associé à leur genre de film

Pour recolter les données necessaire à l'entrainement de l'IA nous nous sommes rendu sur kaggle.com où nous avons recherché un DataSet varié de film associé à leur résumé et leur genre. Cependant avec une plus grosse base de donnée nous aurions eu de meilleur résultat.

Notre base de données a été nettoyé en mettant toutes les descriptions en minuscule. Nous avons remplaçé les abréviations par leur forme développée, par exemple "what's" et "what is". Puis après avoir enlevé les espaces en trop nous avons tokenizer et lemmatizer chaque mot des résumés pour prendre leur racine.

2.2 Base de donnée de champ lexical par genre de film

Pour l'une de nos méthodes de classification nous avons eu besoin d'une base de données de champs lexicaux correspondant au genre des films que nous avons récupéré dans notre première base de données.
Après plusieurs recherches nous avons réalisé que les champs lexicaux dépendaient grandement du contexte du texte donc impossible de trouver une base de données général en ligne.
Nous avons donc décidé d'en créer une nous-même composé de synonyme, des mots les plus fréquents ainsi que de mot choisi arbitrairement représentant le champ lexical de chaque genre.
Une fois les mots de chaque genre sélectionné ils ont subi le même traitement que pour l'autre base de données.
Notre base de données a donc des défauts notamment car nous avons décidé de ne pas y faire figurer certaine catégorie. Les catégorie **documentary**, **reality-tv**, **short** et **talk-show** n'y figure pas puisqu'elles peuvent traiter de tout et n'importe quoi et ne sont donc pas discernable par un vocabulaire.

3 Choix de conception

Plateformes utilisées : **Windows 10** et **Mac OS** (*Montrey*)
Technologie utilisé : **Python3.9+**
Librairies requises :

- pandas==1.3.3 (traitement des données à grande échelle)
- nltk==3.6.6 (tokenisation du texte)
- scikit-learn==1.0 (apprentissage sur des données et prédiction)
- matplotlib==3.4.3 (visualisation de données à l'aide de graphiques)
- argparse==1.4.0 (parsing des entrées du logiciel pour modifier les paramètres)
- tqdm==4.62.3 (visualisation du temps estimé pour effectuer une tâche)

Nous avons fait de la Programmation Orientée Objet pour que le code soit plus facile à utiliser notamment pour entrainer les différents modèles.

4 Implémentation du modèle par champs léxical

- En premier lieu nous nous sommes penchés sur la fréquence des mots pour savoir quelles sont les mots les plus fréquents dans un résumé de film par rapport à son genre.
Par exemple le genre **Horror** :

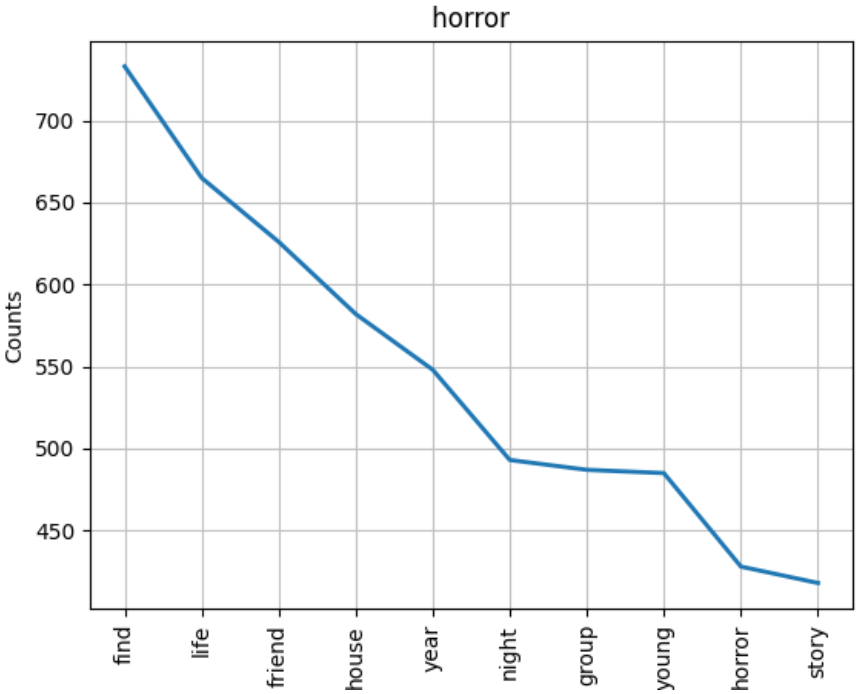

Figure 1 – Horror genre words frequency

- Nous avons ensuite créé une base de données du champ lexical de chaque genre. Maintenant l'idée était de compter le nombre de mots qui appartenait au champ lexical de tous les genres que nous avions dans la base de données.
- De ça nous prenions le genre qui avait le plus de mot appartenant à son champ lexical. Malheureusement cette classification a été un échec total et nous avions un taux d'erreur de 100%.
Notre supposition serait que la base de données pourrait surement être la cause d'autant d'erreurs car beaucoup de mots doivent être manquant et un même mot ce retrouve dans plusieurs genre ce qui fausse les résultats.

5 Comparaison de différents modèle

Nous avons mis en place un comparateur de différents modèle pour voir quel est le plus performant. Pour cela nous avons essayé les 6 models ci-dessous afin de comparer leur accuracy.

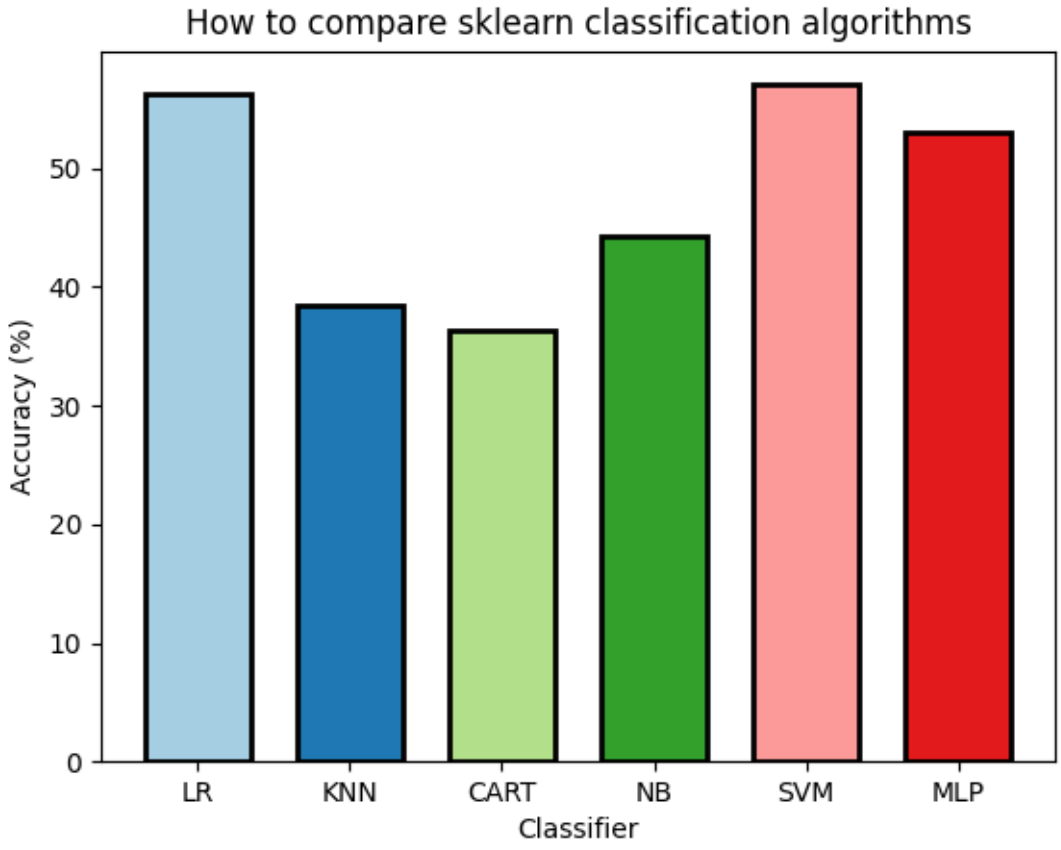

Figure 2 – Comparaison de différents modèles

Les très meilleur semble être SVM, LogisticRegression et MLPClassifier. Les autres étant très éloigné nous avons décidé de tester uniquement ces trois méthodes.

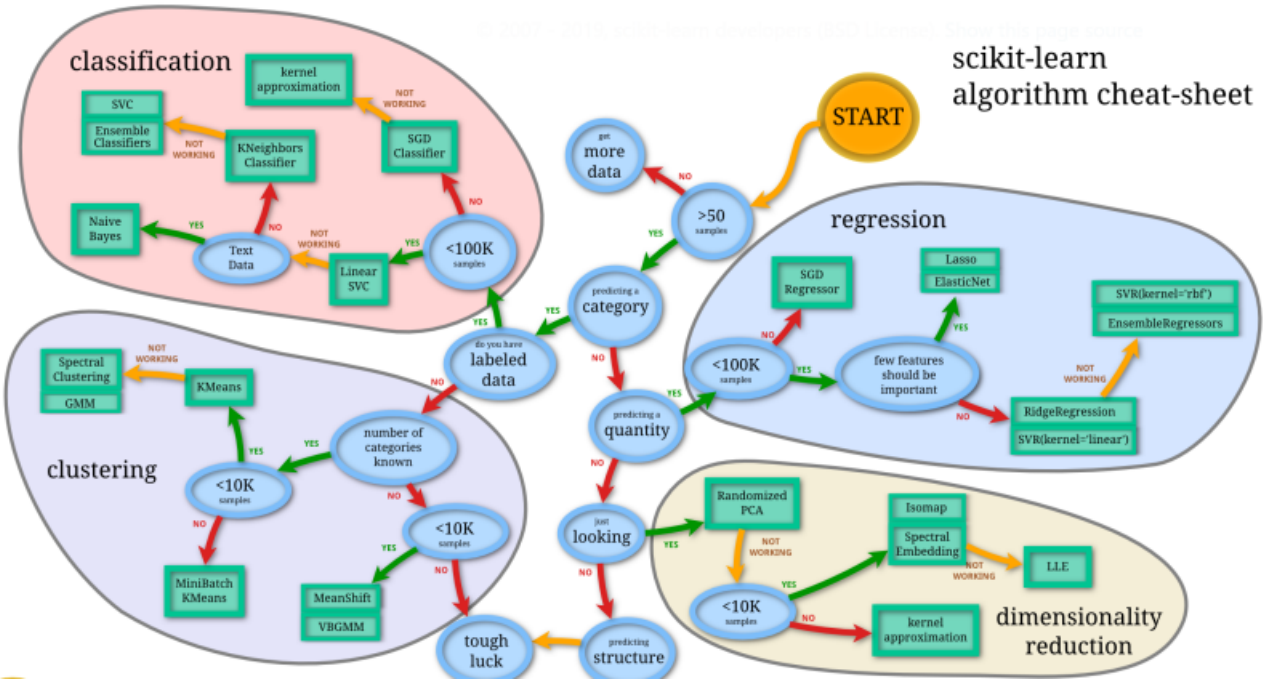

Figure 3 – Roadmap pour le choix d'un modèle

Nous nous sommes aussi appuyés sur l'image ci-dessus qui nous a aidé à choisir le meilleur modèle pour le projet que nous avons à faire en fonction des données. Puisque nous avons une base de données supérieur à 50 et inferieur à 100K et que nous classons du texte dans des catégories. Naturellement l'orientation s'est faite vers la classification avec le modèle LinearSVC.

6 Implémentation de différents modèles

Après avoir réalisé la classification par champs lexical et s'être rendu compte que cette méthode ne fonctionné pas nous nous sommes orientés vers plusieurs classification.

6.1 SVM (Support Vector Machine)

Pourquoi SVM ? Avec plusieurs tests, SVM semble être le plus précis (voir section 5). Les différents paramètres établis pour ce modèle sont test_size=0.1 et train_size=0.5 (voir section 7.1)

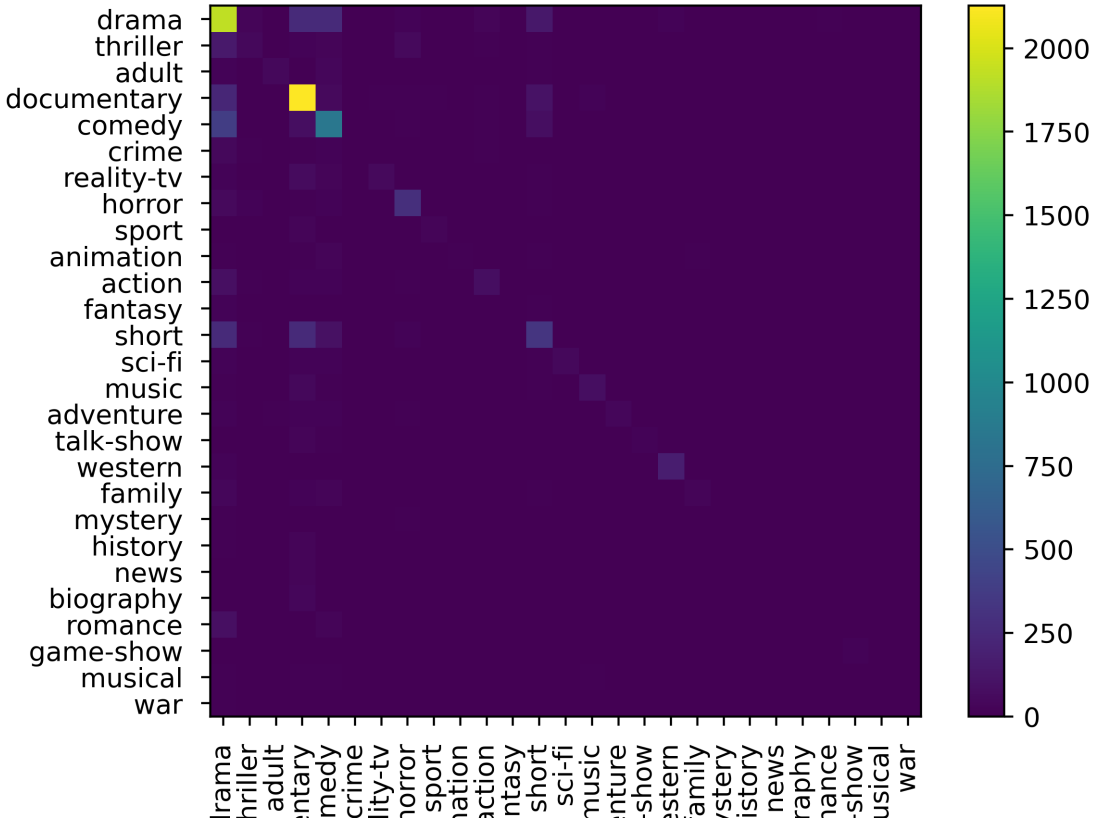

Figure 4 – Matrice de Confusion SVM

Dans la matrice de confusion ci-dessus, il y a **Drama** et **Documentary** qui nous pose problème, ces 2 genres sont énormément confondu avec d'autres.

6.2 LogisticRegression

D'après notre comparaison SVM semblais être le meilleur modèle malgré cela il est très long à entrainer. Comme à ce moment-là nous n'avions pas encore trouvé comment enregistré un model (ce référé section 9) nous voulions essayer une autre méthode. C'est pourquoi nous avons implémenté LogisticRegression.
La LogisticRegression est un algorithme de classification. Il est utilisé pour prédire un résultat binaire basé sur un ensemble de variables indépendantes.

Pour ce modèle nous avons attribué respectivement aux variables test_size et train_size les valeurs 0.2 et 0.7 (ce référé section 7.3). Avec ces paramètres nous avons une accuracy de 58% ainsi qu'un taux d'erreur légèrement supérieur à 42%.

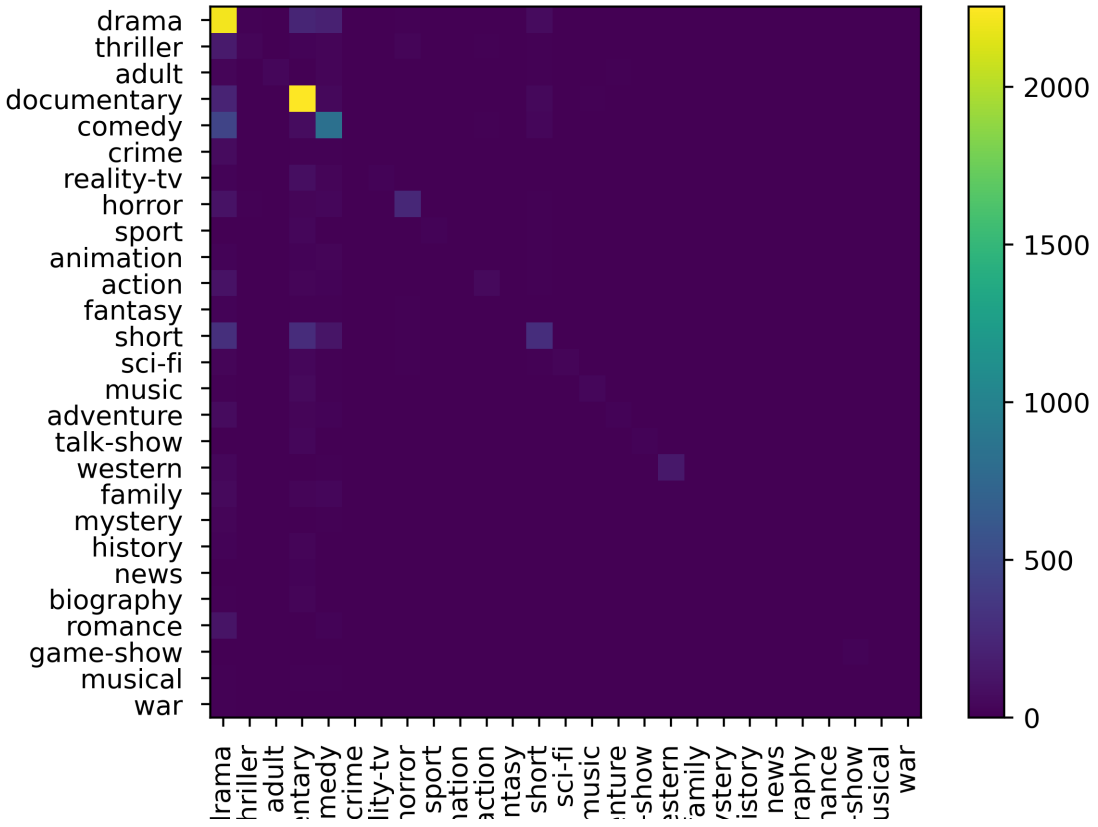

Figure 5 – Matrice de Confusion LogisticRegression

On peut voir que malgré le changement de classification la confusion concernant **Drama** et **Documentary** persiste.

6.3 MLPClassifier

Multi-layer Perceptron Classifier étant un modèle de réseau de neurones nous avons choisi de l'utiliser car l'objectif premier lorsque nous avons commencé le projet était d'en utiliser un pour prédire le genre.

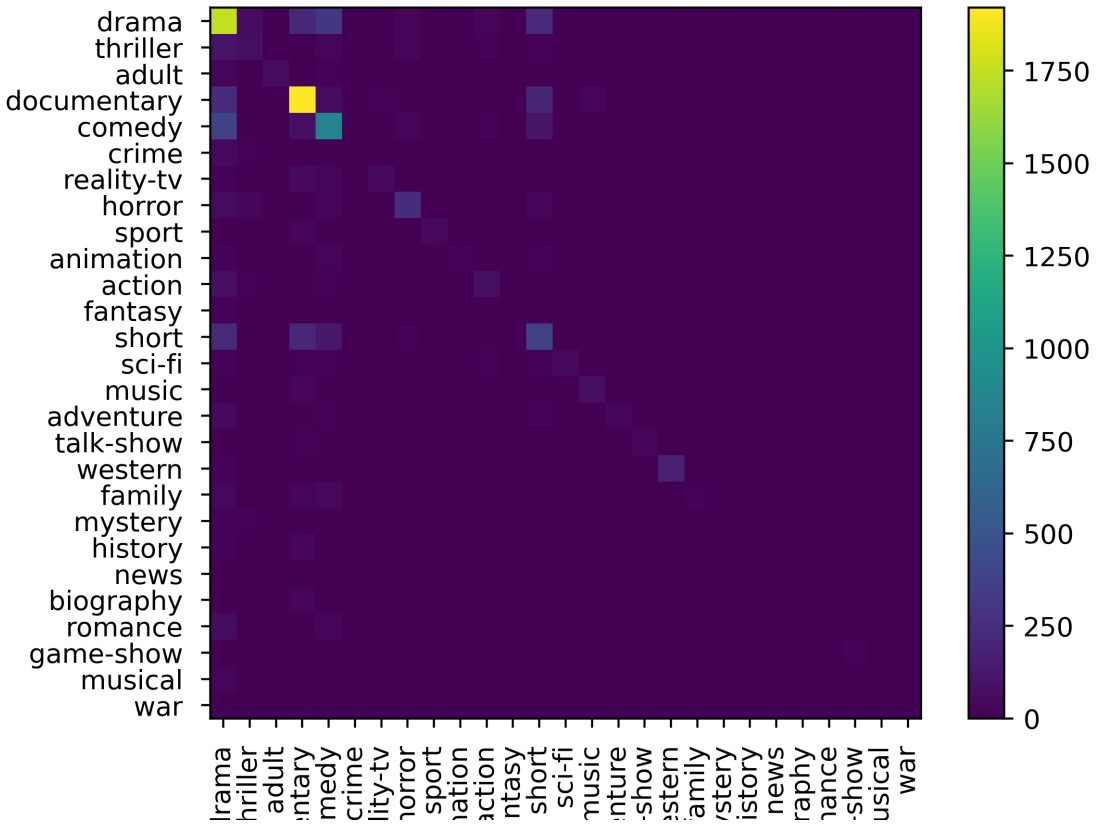

Figure 6 – Matrice de Confusion MLPClassifier

Là aussi pour la matrice de confusion le problème est le même qu'auparavant, nous pouvons tout de même dénoter une accentuation sur le genre **comedy** et une baisse sur **drama**.

7 Tests des modèles par variation de paramètres

Dans le cours de Neural Network and Learning nous avons vue qu'il était important de faire varier les paramètres dans le but de faire augmenter l'accuracy et diminué le taux d'erreur.
Pour chaque méthodes le taux d'erreur a était calculé de manière identique. Nous avons repris les résumés de la base de données initial, les avons classés grâce à la méthode testée puis nous en avons fait un pourcentage d'erreur.
Dans les sections 7.1 et 7.3 nous avons fait varier les paramètres test_size et train_size de 0.1 à 0.9 avec la contrainte que la somme de ces dernières doit être inférieur à 1.

7.1 SVM

Pour illustrer la variation de l'accuracy et du taux d'erreur nous avons réalisé des graphiques. Pour des raisons de lisibilité nous avons fait plusieurs graphiques, vous pouvez donc voire ici le graphique comportent le meilleur résultat que nous ayons obtenue pour SVM, 58% d'accuracy pour 43% d'erreur, pour test_size = 0.1 et train_size = 0.5.

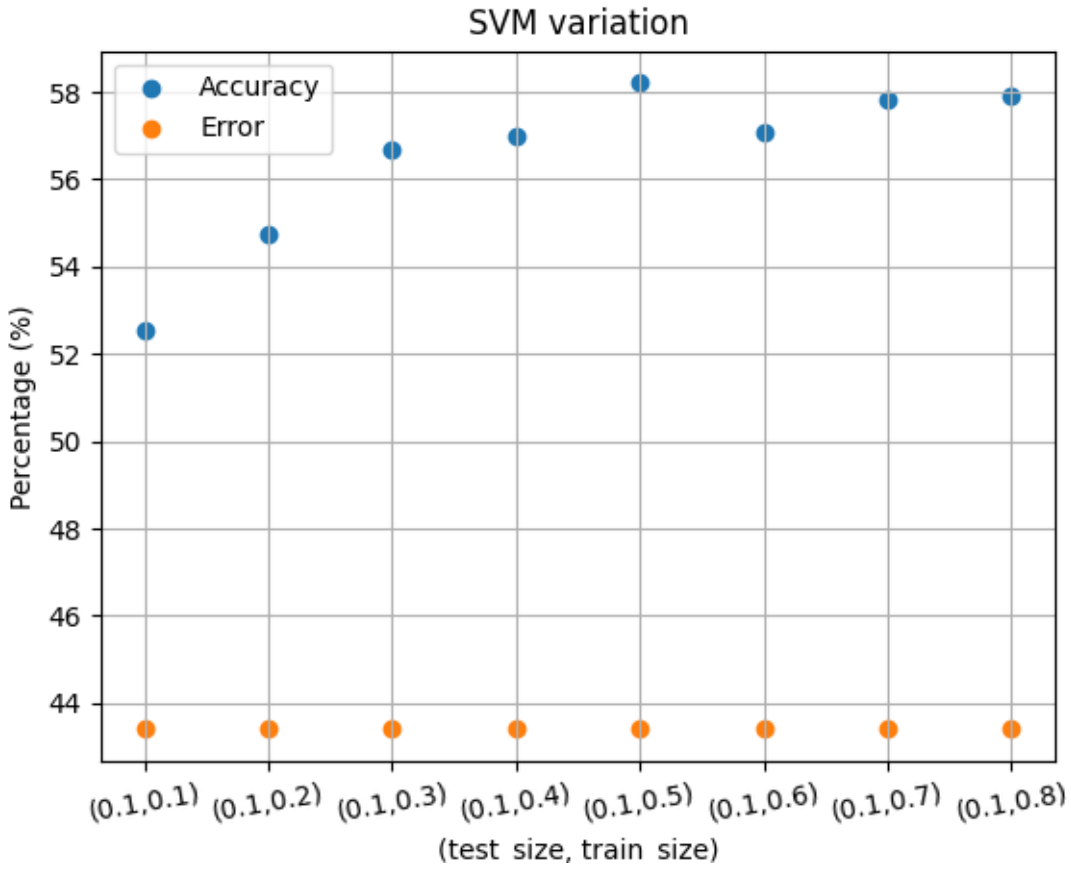

Figure 7 – SVM variation

7.2 LogisticRegression

Vous pouvez également voire ici le graphique comportent le meilleur résultat que nous ayons obtenue pour LogisticRegression, 58% d'accuracy pour 42% d'erreur, pour test_size = 0.2 et train_size = 0.7.

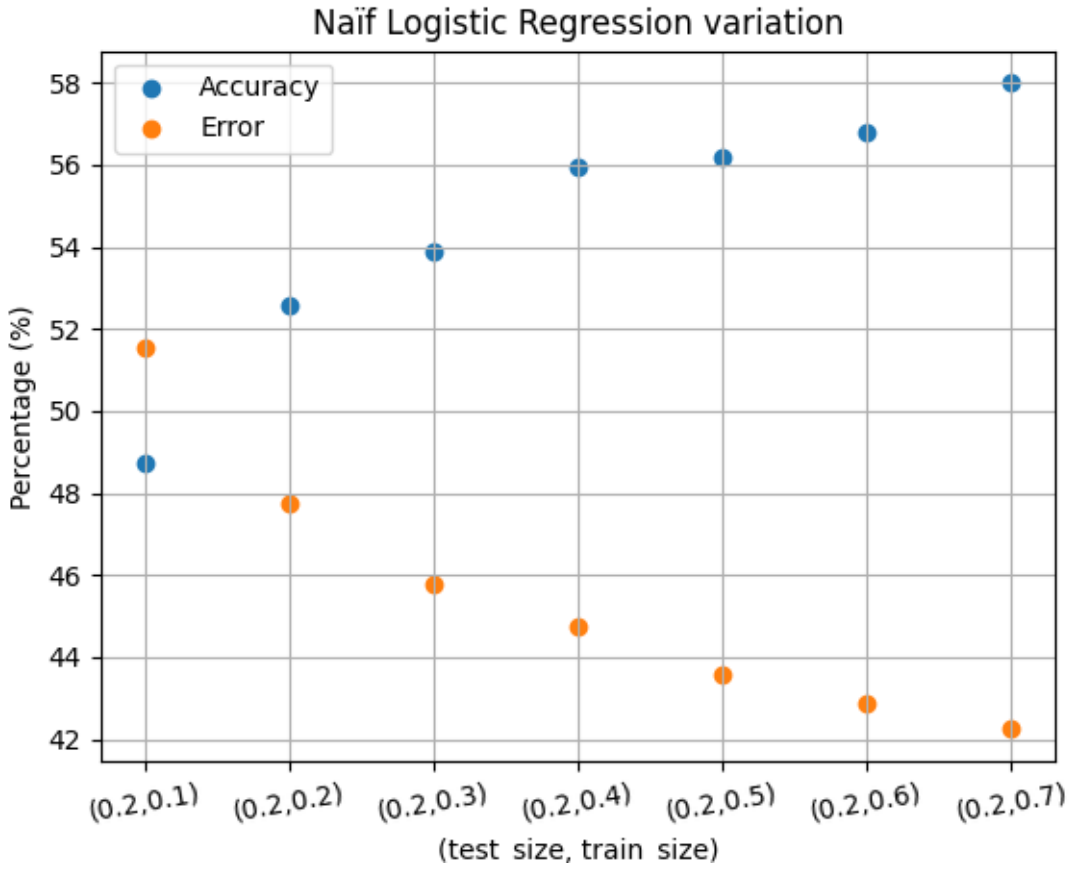

Figure 8 – LogisticRegression variation

7.3 MLPClassifier

MLP étant un modèle de réseau de neurones nous avons fait varier le nombre de couche ainsi que le nombre de neurones par couche.
Comme pour les graphiques précédents on voit uniquement celui comportant le meilleur résultat, 55% d'accuracy pour 46% d'erreur, pour 1 couche composée de 46 neurones.
Nous avons décidé de prendre test_size=0.2 et train_size=0.7 comme LogisticRegression car nous n'avons pas eu le temps de faire les tests pour chaque paramètre.

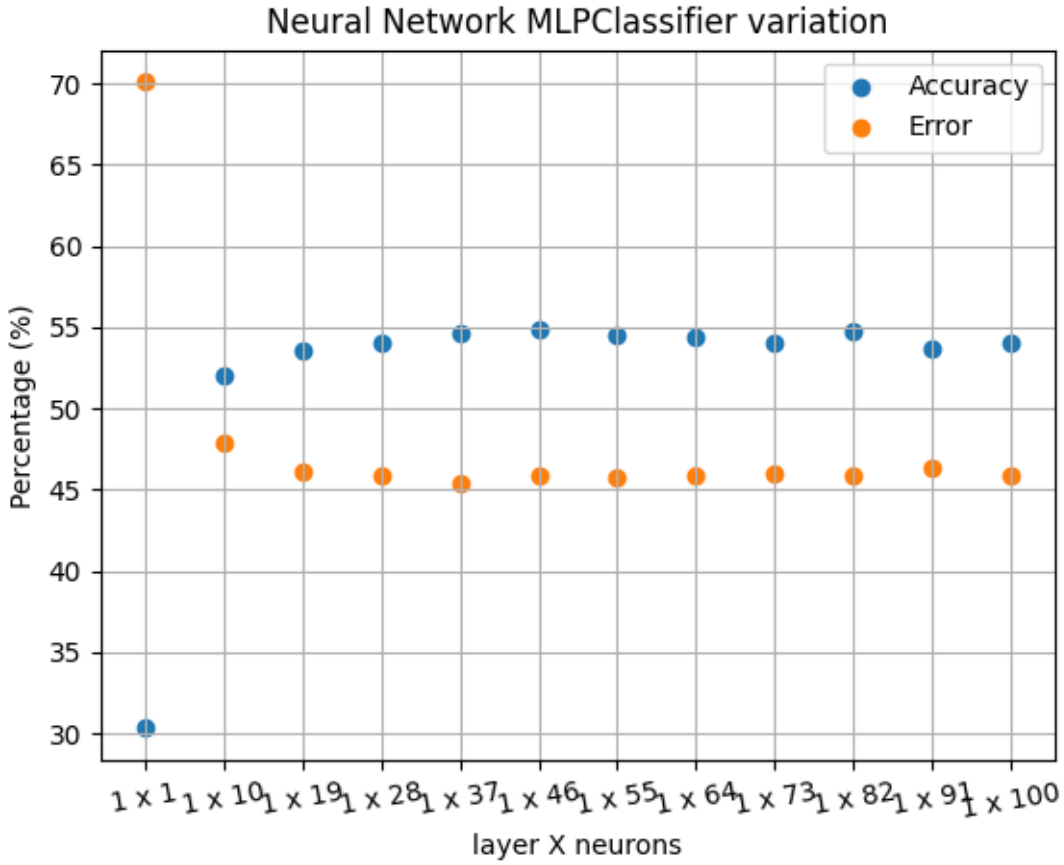

Figure 9 – MLPClassifier variation

8 Comment utiliser

Il y a deux possibilités pour lancer le programme soit avec Jupyter ou en ligne de commande. Jupyter n'est là que pour prédire un exemple avec les modèles existant et ne pourra pas entrainer un modèle. Si vous souhaitez tester un film différent avec un modèle différent ou entrainer un modèle avec des paramètres plus précis il faudra passer par les lignes de commandes dans le terminal.

Pour plus d'informations allez sur le README.md du projet ou faite la commande : **python3 classifier.py -h**

8.1 Prédiction avec modèle préexistant

Par exemple si l'on veut prédire le genre du film Dune vous pouvez le faire avec la commande ci-dessous :

Input :
python3 classifier.py -cl svm -mp models/svm.sav -t "Dune" -d "The story of Paul Atreides, a young man as gifted as he was brilliant, destined to meet an extraordinary destiny that totally surpasses him. Because if he wants to preserve the future of his family and his people, he will have to go to the most dangerous planet in the universe - the only one able to provide the most precious resource in the world, capable of multiplying tenfold. power of mankind. As evil forces vie for control of this planet, only those who manage to overcome their fear will be able to survive ..." -csv dune_predicted.csv

Output :

```
% Predicting with LinearSVC()...
Prediction done and saved in dune_predicted.csv
Classifier LinearSVC() prediction for ['Dune'] is [' sci-fi ']
```

8.2 Apprentissage

Si l'on souhaite entrainer un modèle nous pouvons utiliser la commande ci-dessous :

Input :
python3 classifier.py –train true -mp "svm.sav" -cl svm -trs 0.5 -tes 0.1

Output :

```
% Start training with <class 'sklearn.svm._classes.LinearSVC'>...
svm.sav saved
Trained with 57.064% accuracy
```

9 Sauvegarde et chargement de modèles pré-entrainés

Pour éviter le temps d'attente lorsque nous entrainons notre modèle nous avons mis en place une sauvegarde du modèle grâce a la librairie pickle qui enregistre des objets python en binaire et qui nous permet de charger un modèle préexistant sans devoir l'entrainer à chaque reprise.

10 Améliorations possible

Nous avons pensé à plusieurs améliorations possibles :

- Trouver un meilleur jeu de donnée pour que nos résultats soit plus précis
- Utiliser plus spécifiquement chaque paramètre des modèles avec d'avantage de tests notamment les paramètres test_size et train_size du model MLPClassifier
- Classer les résumés dans plusieurs genre ou sous-genre
- Reconnaitre le genre des films peu importe la langue du résumé
- Trouver une base de données adapté pour classer les films grâce à leur scénario

11 Conclusion

Nous avons pu étudier différentes méthodes d'approche pour classifier un résumé de film associé à un genre. Pour parvenir à trouver un résultat satisfaisant il a fallu passer par une multitude de tests et de lecture d'articles pour savoir ce qui pourrait être le plus adapté à notre projet.

Parmi toutes les méthodes de classification testée nous retenons que **MLPClassifier**, **SVM** et **LogisiticRegression** car ce sont les seuls qui nous donnent des résultats positifs. Tandis que la méthode par champ lexical avec un taux d'erreur de 100% n'est pas du tout viable.

Au départ nous aurions pensé que le réseau de neurones, **MLPClassifier**, aurait était le meilleur mais au vu des résultats nous en concluons que **SVM** et **LogisticRegression** sont meilleur.
Nous aurions dû tester de faire varier test_size et train_size pour le réseau de neurones mais par manque de temps nous n'avons pas pu. **SVM** et **LogisticRegression** sont équivalent puisque **SVM** a une accuracy légèrement plus élevé et **LR** un meilleur taux d'erreur mais de peu.

N.B : Vous pouvez retrouver tous les graphiques dans l'archive de notre projet.