

Projet Fil Rouge ML

Yousra Bouchikhi et Joanne Couallier

1 Introduction

L'objectif de ce projet est de reconnaître des enregistrements audio de commandes d'un drone quadri-coptère. Nous avons décidé d'entraîner nos modèles sur les données d'enregistrements d'étudiants, complétées par les enregistrements effectués par les étudiants de notre classe. Puis, une fois les modèles entraînés, nous pourrons les évaluer sur les enregistrements d'étudiants étrangers et observer les résultats.

2 Pré-traitement des données

2.1 De l'audio aux données

Après le téléchargement des fichiers audio à l'aide de la librairie librosa, nous supprimons les blancs des audios tels que le silence au début et à la fin de chaque audio. Ensuite, nous ramenons tous les audios à la même durée qui représente la durée moyenne de tous les audios constituant notre base de donnée. Nous avons fait ce dernier choix afin que le nombre de fenêtres par segment soit le même pour tous les audios et donc la quantité d'informations extraite de chaque audio soit la même pour tous.

On extrait ensuite de chaque audio un certain nombre de coefficients MFCC, 12 coefficients dans notre cas (d'autres valeurs ont été testée et les résultats se trouve dans le notebook). Une fois les coefficients MFCC extraits, la fonction ReductionMemeDimension, comme l'indique son nom, réduit les fichiers audio à la même dimension.

Le signal audio est d'abord divisé en $Nb_Segments$ segments. Puis chaque segment est divisé en un nombre égales de fenêtres qui vaut (**taille d'un coefficient MFCC / Nb_Segments**), dans notre cas, la taille d'un coefficient MFCC vaut 35 et le nombre de segments vaut 2 donc le nombre de fenêtres par segment vaut 17 (on prend le premier entier inférieur au résultat de la division précédente). Puis nous calculons la moyenne des valeurs de chaque coefficient MFCC sur chaque fenêtre d'un segment pour sur chaque segment. La fonction retourne alors une matrice de taille (**le nombre de coefficients MFCC choisi , Nb_Segments**). Nous avons fait le choix de conserver 2 segments par signal audio, puisqu'ils contenaient assez d'informations sur le signal pour bien entraîner nos modèles, et qu'un nombre de segments supérieur à 2 conduisait à un sur-apprentissage des modèles.

2.2 Choix des labels

Les données sont ensuite labélisées, on retrouve les 14 labels suivants (numérotés de 0 à 13): *tourne droite, tourne gauche, avance, recule, gauche, droite, arrête toi, atterrissage, décollage, état d'urgence, fais un flip, plus bas, plus haut, pendule inverse*.

Mais ils ne sont pas tous présents avec la même fréquence.

Nous disposons de trop peu de données, par rapport au nombre de labels possible. Pour choisir les labels sur lesquels nous allons entraîner les algorithmes, nous cherchons ceux qui se ressemblent le moins. En effet, si deux enregistrements ont des mfcc très similaires et des labels différents, l'apprentissage des modèles ne sera pas très correcte. On peut par exemple observer le graphe des 12 boxplots des coefficients MFCC pour chaque label (Figure 1).

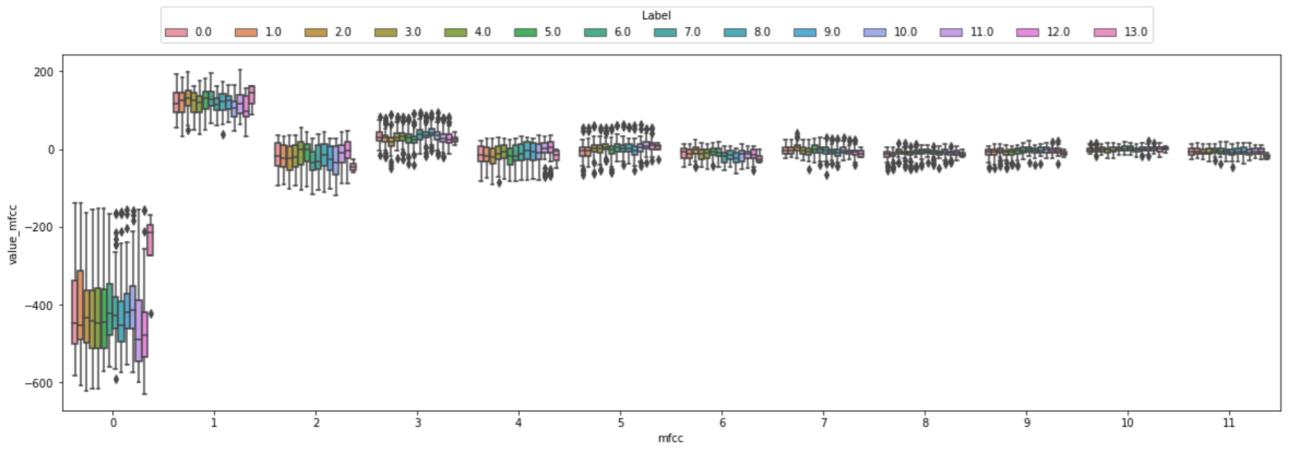


Figure 1: Boxplot des coefficients MFCC par labels

Pour quantifier l'écart entre les boxplots, on peut calculer, pour chaque MFCC, les distances de chaque label par rapport aux autres et les afficher graphiquement grâce à une heatmap (voir dans le notebook). On affiche dans le notebook la matrice des distances en excluant le label 13 ("pendule inverse" qui est très peu observé) pour avoir une échelle de couleurs plus adaptée. L'objectif est de choisir des labels tels que leurs distances à tous les autres labels est grande.

- on peut d'abord choisir un label parmi 11 et 12 qui sont très proches entre eux, mais très éloignés de tous les autres
- on choisit ensuite les labels 5, 6 qui sont relativement éloignés l'un de l'autre ainsi que des labels 11, 12.

On propose donc de garder : **droite** (= 0), **arretetoi** (= 1), **plushaut** (= 2).

2.3 Construction Base de données d'apprentissage et de test

Pour la construction de nos bases de données d'apprentissage et de test, on choisit de garder 80% des données pour l'apprentissage et 20% pour le test, sans remise. On fait aussi en sorte de garder les mêmes proportions de chaque label. C'est à dire, que pour chaque label, on tire aléatoirement 80% des données pour l'apprentissage et 20% pour le test, et on continue sur les labels suivants.

Enfin, nous avons aussi construit d'autres bases de données sur lesquelles on pourra essayer de faire tourner les modèles : la base des données normalisées et la base des données réduites par ACP.

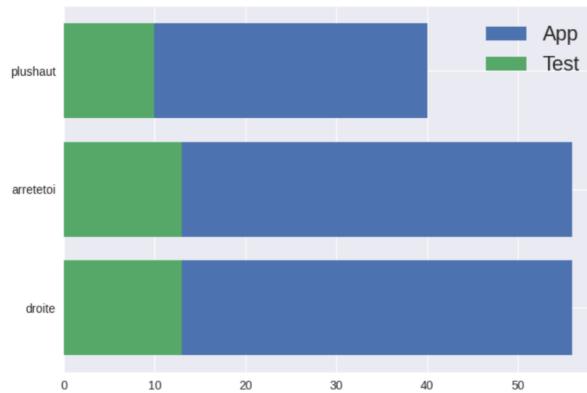


Figure 2: Constitution des Bases App et Test

3 Régression Multiclasse

L'algorithme peut utiliser une méthode binomiale ou multinomiale. On fait ensuite une régression logistique régularisée.

3.1 Choix du modèle

On peut faire varier les paramètres suivants :

- l'optimiseur : *lbfgs*, *saga*, *sag*, *newton-cg*
- C : coefficient de régularisation, plus C est petit, plus la régularisation est forte

Les paramètres sélectionnés sont : optimiseur = 'lbfgs' et C = 2.0

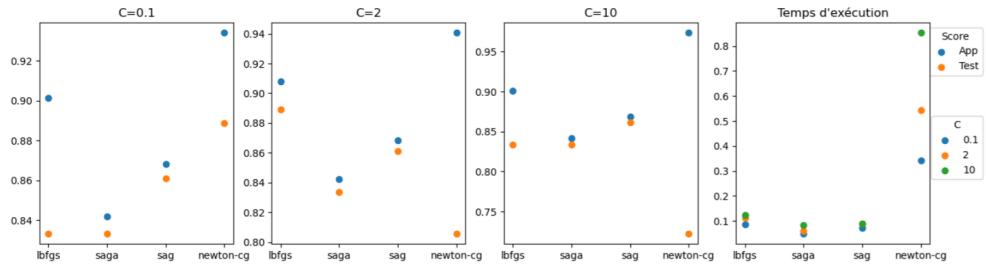


Figure 3: Comparaison des scores d'apprentissage et de test et des temps d'exécution

En faisant varier ces deux paramètres on obtient les graphes de la figure(3). Une fonction nous permet ensuite de garder les paramètres donnant les meilleurs résultats. C'est-à-dire des paramètres pour lesquels le score de test est le plus élevé possible, avec un écart entre score de test et score d'apprentissage le plus faible possible, pour éviter le sur-apprentissage.

On remarque aussi que le choix de l'optimiseur influe sur le temps d'exécution, qui est inférieur pour les optimiseurs 'lbfgs', 'saga' et 'sag'.

Le modèle binomiale optimisé donne 0.8947 en score d'apprentissage et 0.8888 en score de test. Contre 0.9013 en score d'apprentissage et 0.8333 en score de test pour le modèle multinomiale. On va donc garder un modèle multinomiale.

3.2 Évaluation des résultats

Pour évaluer les résultats, on considère le score obtenu, les courbes ROC de chaque label et la matrice de confusion.

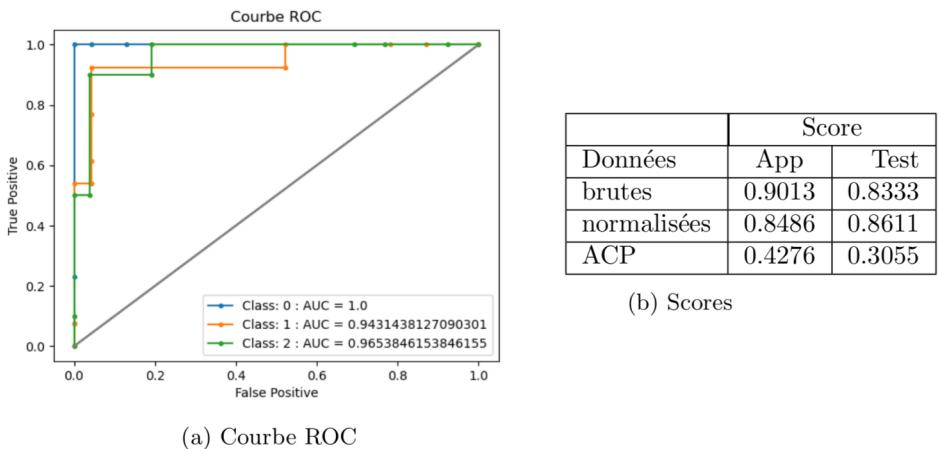


Figure 4: Evaluation des résultats de la Régression Multinomiale

La courbe ROC permet de visualiser le taux de mauvais classement et de bon classement. La diagonale représentant un choix aléatoire de classification. Lorsque les courbes sont au dessus de la droite (AUC) et s'approchent de 1.0, meilleure est la classification. Le taux de vrais positif doit être supérieur au taux de faux positifs. Ici Les trois courbes sont bien proche de l'angle à 45 degrés.

La figure 4b donne les scores pour les modèles optimisés sur chaque base de données. On remarque que les données réduites par ACP donnent des résultats très insatisfaisants, la réduction a fait perdre trop d'informations. Tandis que les scores sur les données normalisées sont bons, mais le modèle semble faire du sur-apprentissage puisque le score de test est meilleur que celui de l'apprentissage. Les données brutes donnent donc les meilleurs résultats sans pour autant induire à un sur-apprentissage.

4 Classification par méthodes à noyau

Les méthodes à noyau consistent à plonger les données dans un espace de dimension de Hilbert \mathcal{H} où les données pourront être séparés linéairement.

4.1 Classification par SVM à noyau

On utilise la fonction SVC pour Support Vector Classification. La classification multi-classe est supportée en considérant chaque classe contre les autres.

4.1.1 Choix du modèle

On peut faire varier les paramètres suivant :

- le kernel : *linear, poly, rbf, sigmoid, precomputed*
- C : coefficient de régularisation

Les paramètres sélectionnés sont : kernel = 'poly' et C = 15

4.1.2 Évaluation des résultats

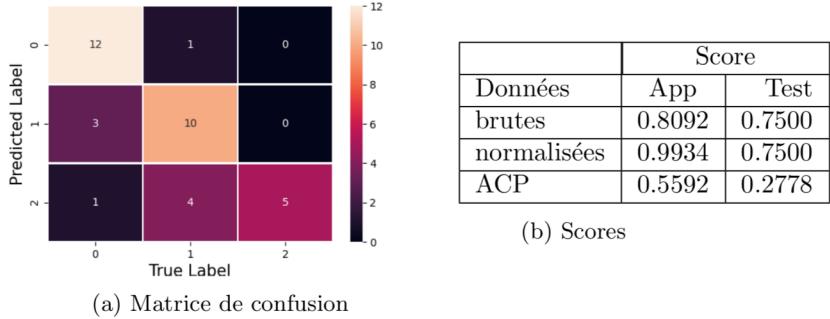


Figure 5: Evaluation des résultats de la SVM

La matrice de confusion représente le nombre de labels bien prédits, et de labels mal prédits. Un bon modèle va donc donner une matrice avec des coefficients élevés sur la diagonale, et très faibles voir nuls ailleurs. Ici on obtient plutôt de bons résultats.

4.2 Approche non supervisée

Dans cette approche, on ne connaît pas les labels. Le modèle va séparer les enregistrements en plusieurs classes, mais ne pourra pas dire s'il s'agit de la classe 'arretetoi', 'droite' ou 'plushaut'.

Choix du modèle : On peut faire varier la méthode d'assignation des classes ('discretize' ou 'kmeans'). Ici on va plutôt garder 'discretize'. On conserve 'lobpcg' comme eigen_solver (Stratégie de décomposition en valeurs propres). Mais cette méthode donne de très mauvais résultats. Ce qui était prévisible, puisqu'elle est non-supervisée.

Évaluation des résultats

		Score	
Données	App	Test	
brutes	0.3487	0.3611	
normalisées	0.3553	0.3889	
ACP	0.3552	0.3333	

5 Apprentissage par ensemble : AdaBoost, Gradient Boosting

5.1 AdaBoost

Ce modèle commence par effectuer une classification sur le dataset original, puis entraîne des copies du classifieur sur le même dataset, mais en assignant des poids aux enregistrements mal classés. On peut le voir comme une succession de modèles, où le premier modèle fait une prédiction globale, puis les suivants corrigent les erreurs du précédent.

5.1.1 Choix du modèle

On peut faire varier les paramètres suivant :

- le nombre maximum d'estimateurs $\in [1, +\infty[$
- le learning rate : $\in [0, +\infty[$ c'est le poids assigné à chaque classifieur, à chaque itération du boosting. Une valeur élevée de lr augmente la contribution de chaque classifieur.

Les paramètres sélectionnés sont : n_best = 100 et LR = 2.90.

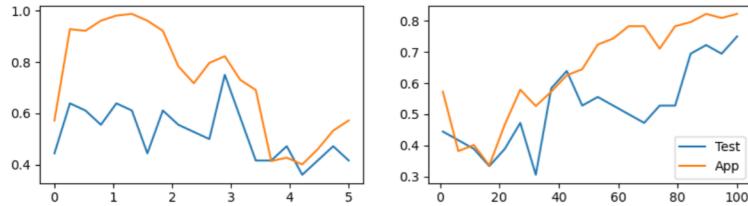


Figure 6: Score en fonction des paramètres n_est et LR

Lorsque le nombre d'estimateurs augmente, globalement le score augmente aussi. Ici l'objectif est de sélectionner les paramètres tel que le score est le plus élevé, mais aussi tel que les courbes d'apprentissage et de test soient les plus proches possible.

5.1.2 Évaluation des résultats

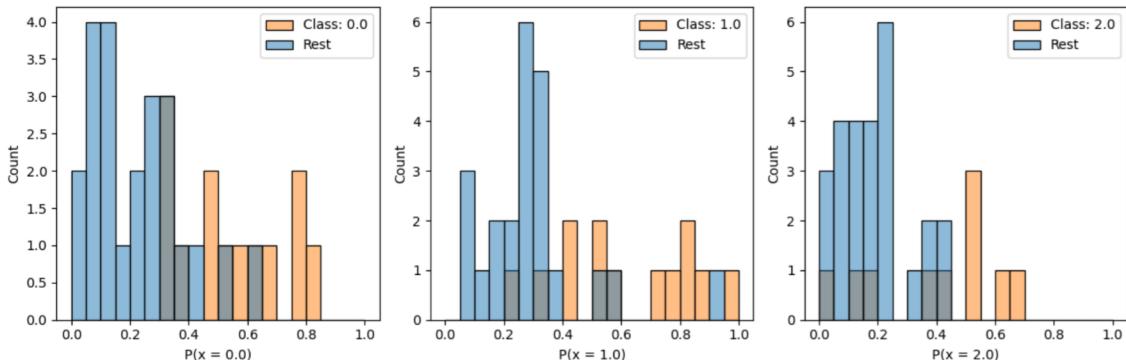


Figure 7: Distribution des classes

Plus les distributions bleues et oranges sont séparées, meilleure est la classification. Ici les résultats sont plutôt correctes. On obtient des scores de 0.8224 en apprentissage et 0.75 en test pour le modèle sélectionné.

5.2 Gradient Boosting

Le Gradient Boosting regroupe une descente de gradient et un boosting. Ici on minimise la fonction perte. Le boosting permet de combiner plusieurs weak learners pour faire un strong learner. On entraîne les modèles séquentiellement, pour corriger le modèle précédent.

5.2.1 Choix du modèle

On peut faire varier les paramètres suivant :

- **le learning rate** : donne la contribution de chaque arbre ($\in [0, +\infty[$)
- **Profondeur maximale** (`max_depth`) : limite le nombre maximal de noeuds dans chaque arbre. La valeur optimale dépend de l'interaction entre les variables d'entrée. ($\in [1, +\infty[$)

Les paramètres sélectionnés sont : LR = 0.01 et `max_depth` = 2.

Le score se stabilise pour un learning rate supérieur à 0.2.

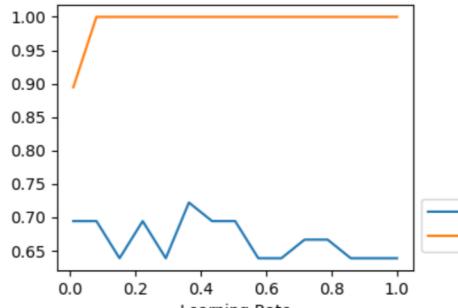
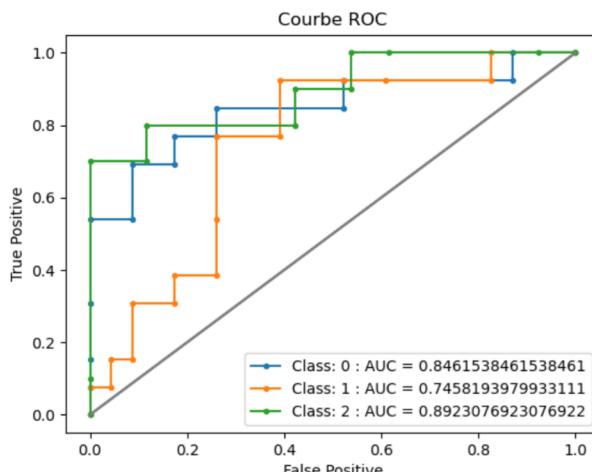


Figure 8: Score en fonction du learning rate

5.2.2 Évaluation des résultats



(a) Courbe ROC

	Score	
Données	App	Test
brutes	0.8947	0.6944
normalisées	0.8947	0.6944
ACP	0.6184	0.4166

(b) Scores

Figure 9: Évaluation des résultats de Gradient Boosting

Remarques : Les classes 0 et 2 sont plutôt bien prédites, ce n'est pas le cas pour la classe 1 qui s'approche de la droite (AUC). Plus ces quantités sont proches de 1, meilleure est la classification.

Cependant, il y a un écart important entre les scores d'apprentissage et de test. Le modèle de Gradient Boosting fait du sur-apprentissage. Le modèle a un très bon score sur les données d'entraînement, mais prédit mal sur les données de test. Et ceci pour les 3 bases de données : brute, normalisée et réduite avec ACP. Ce modèle n'est donc pas exploitable dans le cadre de notre étude.

6 Classification par réseaux de neurones

On utilise ici un Multi-layer Perceptron classifier. C'est un type de réseau neuronal formel qui s'organise en plusieurs couches. Chaque couche se compose d'un nombre de neurones variable. Les neurones de la dernière couche sont les sorties du système global. Ce modèle optimise la log-loss.

6.1 Choix du modèle

On peut faire varier les paramètres suivant :

- **activation** : la fonction d'activation pour les hidden layers : 'identity', 'logistic', 'tanh', 'relu'
- **solver** : le solver pour l'optimisation des poids: 'sgd', 'adam'.
- **alpha** : force de la régularisation L2.

Les paramètres sélectionnés sont : activation = 'identity', solver = 'adam' et alpha = 0.01.

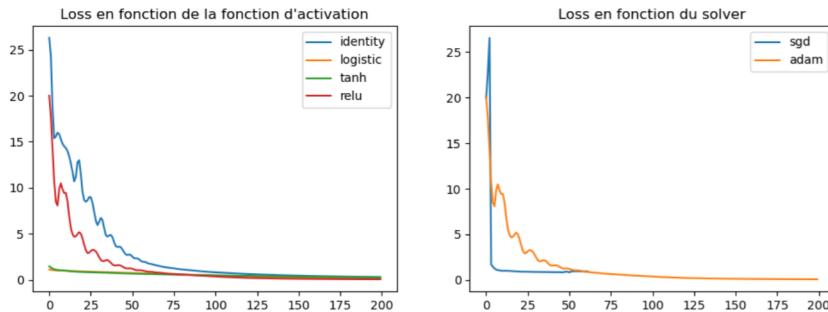
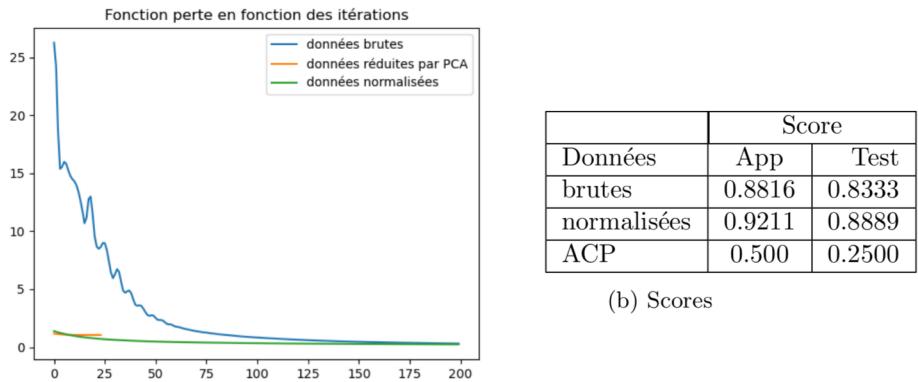


Figure 10: Fonctions pertes en fonction des paramètres d'activation et du solver

Ici on observe que les courbes loss décroissent rapidement. 'sgd' semble converger plus rapidement que 'adam', mais 'adam' atteint une loss plus faible. Concernant l'activation, 'tanh', 'relu' ont l'air de converger beaucoup plus rapidement est pour une valeur finale équivalente. Mais lorsque l'on calcule les scores sur les données de test, ces paramètres entraînent de l'over-fitting.

6.1.1 Évaluation des résultats



(a) Courbe Loss des données d'apprentissage

Figure 11: Évaluation des résultats de MLPClassifier

Remarques : Les scores sont bons pour les données normalisées et brutes mais très mauvais pour les données réduites avec ACP(sur-apprentissage). Les scores sont meilleurs sur les données normalisées, avec un écart entre le score d'apprentissage et de test légèrement plus faible que sur les données brutes. On retrouve cela sur les courbes loss. La courbe des données normalisées (en vert) converge plus rapidement.

7 Comparaison des Modèles

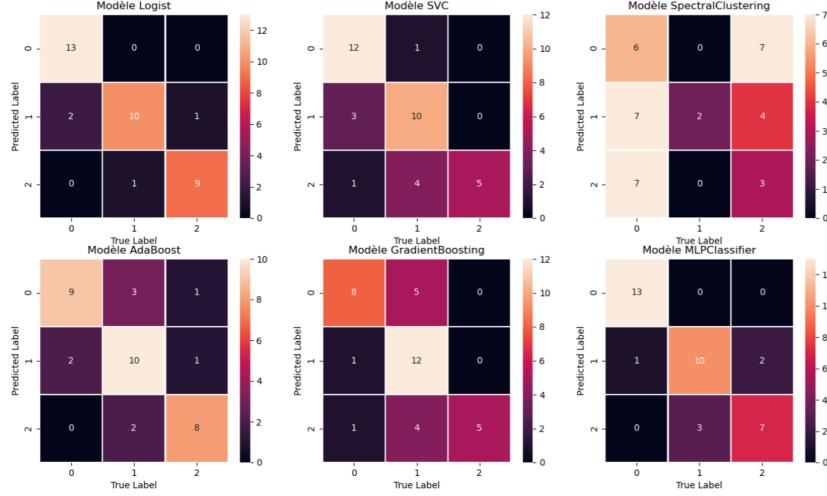


Figure 12: Comparaison des matrices de confusion

Remarques : Les matrices de confusion (sur la base de test) permettent de comparer visuellement les résultats des différents modèles. Les meilleurs modèles ont une diagonale avec des couleurs claires et des couleurs foncées partout ailleurs. Ici on retrouve bien que la régression logistique et les réseaux de neurones multicouches donnent de meilleurs résultats, avec des résultats plutôt satisfaisants par AdaBoost.

On compare les scores obtenus par les différents modèles sur la base de donnée non normalisée et non réduite.

Modèles	Score App	Score Test	Remarques
Régression Logistique	0.9079	0.8889	Bons résultats, pas d'overfitting
SVM	0.8092	0.7500	moins bon mais correct
Spectral Clustering	0.2961	0.3056	mauvais résultats, légèrement meilleurs sur les données normalisées
AdaBoost	0.8224	0.7500	Bons résultats, moins bon pour la classe 1
Gradient Boosting	0.8947	0.6944	problème d'overfitting
MLPClassifier	0.8816	0.8333	Bons résultats, meilleur sur données normalisées

Commentaires :

- Les meilleurs résultats d'apprentissage sont obtenus avec la régression logistique et le Gradient Boosting.
- Cependant le gradient boosting fait de l'over-fitting, le score de test est bien plus faible.
- La régression logistique et les réseaux de neurones (MLPClassifier) donnent des résultats d'apprentissage et de test très rapprochés et très satisfaisants compte tenu de la taille de notre base de données.
- Le modèle Spectral Clustering n'est pas adapté à notre étude puisque c'est un modèle de classification non supervisée, ce qui est reflété sur ses scores d'apprentissage et de test.

8 Évaluation sur de nouvelles données

Nous allons maintenant tester nos modèles préalablement entraînés, sur une base de données d'audios enregistrés par des étudiants étrangers pour évaluer leur performance sur de nouvelles données.

8.1 Prétraitement des nouvelles données

De la même manière que la base de données d'entraînement, nous appliquons nos transformations à la base de données des étudiants étrangers en supprimant les blancs des audios et les ramenant à la même longueur suivi par une extraction des coefficients MFCC et du calcul des moyennes des valeurs de chaque coefficient MFCC sur des fenêtres de même taille par segment.

8.2 Évaluation des meilleurs modèles sur les nouvelles données

L'évaluation des différents modèles retenus après l'étude effectuée plus haut donne les résultats suivants:

Modèles	Score Test	Commentaire
Régression Logistique	0.6667	bon score
SVM	0.6363	pas adapté
AdaBoost	0.4848	satisfaisant mais pas le meilleur modèle
Gradient Boosting	0.5454	sur-apprentissage
MLPClassifier	0.7575	bon score

On peut aussi remarquer que ces scores sont inférieurs à ceux obtenus sur notre base de donnée d'apprentissage, ce qui est normal puisque les modèles ne sont pas entraînés à reconnaître les voix et accents des étudiants étrangers mais plutôt des enregistrements vocaux d'étudiants français.

Les modèles arrivent quand même à trouver des ressemblances entre la base de donnée d'évaluation et celle d'apprentissage.

D'après les scores de test ci dessus nous pouvons dire que les modèles qui prédisent le mieux les classes des commandes vocales par des étudiants étrangers sont : la régression logistique et les réseaux de neurones, ce qui est cohérent avec l'étude effectuée plus haut.

NB: Nous avons supprimé le modèle Spectral Clustering puisqu'il ne donne pas de résultats satisfaisants et donc n'est pas adapté dans le cadre de notre étude.

Les différentes matrices de confusion représentant les prédictions de ces modèles sur la base de données d'évaluation sont les suivantes:

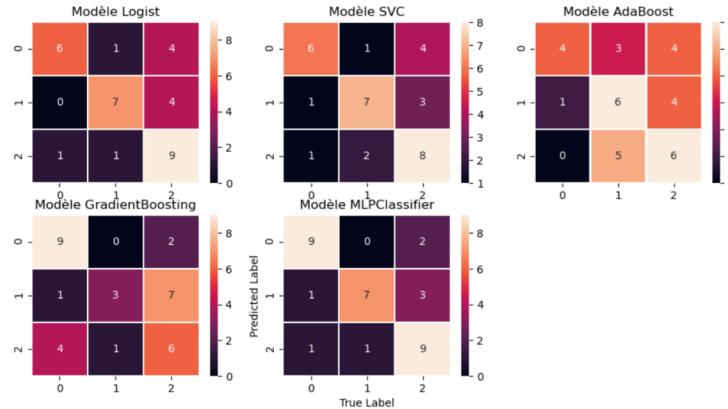


Figure 13: Comparaison des matrices de confusion sur la base de données Etudiants étrangers

En effet, nous pouvons remarquer que les modèles qui classifient le mieux les labels de la base de données d'évaluation sont la régression logistique et les réseaux de neurones.

9 Bilan

Lors de l'étude, nous avons remarqué l'importance du pré-traitement des données, qui a nettement permis d'augmenter les scores obtenus. Pour cela il a fallut comprendre la structure des données et les défauts qu'elles pouvaient avoir :

- blanc en début et fin d'enregistrements : informations inutiles.
- audios de durées différentes
- ressemblance entre certains labels
- fréquence des labels différentes, etc

Il est aussi important d'avoir le plus de données possibles. Surtout si l'on souhaite considérer plus de 3 labels. Ici on a dû mélanger des bases de données différentes pour avoir suffisamment d'enregistrements. Il est également important d'être vigilant à la construction de nos données d'apprentissage et de test. En évitant d'avoir des éléments qui se répètent dans les deux bases. Il faut aussi essayer au maximum d'avoir une répartition équitable des labels pour l'apprentissage.

Une fois que nos bases d'apprentissage et de test sont construites, les modèles sont optimisés avec les paramètres les plus adéquats à notre étude, et on peut analyser les résultats grâce à différentes métriques, tel que les scores, les courbes ROC, les matrices de confusion, les fonctions pertes.

L'étude pourrait maintenant être poursuivie en prenant en compte plus de 3 labels, ou en utilisant d'autres modèles, comme les forêts aléatoires...

On pourrait aussi exploiter d'autres méthodes de traitement du son comme la suppression du bruit. Et trouver d'autres méthodes pour extraire l'information de nos enregistrements que les coefficients de Mel.