

---

# Rapport PROJET S7

RecognAItion - reconnaissance d'émotions dans la voix

---

*Réalisé par :*  
Alexandre GASTINEL  
Paul LEMAIRE  
Nathan LEWY  
Pierre SAINCTAVIT

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contexte et objectif	1
1.2	Etat de l'art	1
1.2.1	Présentation des bases de données	1
1.2.2	Présentation de modèles sans deep learning	1
1.2.3	Présentation de modèles avec deep learning	2
<b>2</b>	<b>Préparation des données</b>	<b>3</b>
2.1	Introduction	3
2.2	Caractéristiques Exploitées	3
2.3	Une Première Solution avec <i>OpenSMILE</i>	4
2.4	Optimisation du choix des features avec l'Information Mutuelle	5
2.4.1	Définition de l'Information Mutuelle	5
2.4.2	Défis liés à l'utilisation de l'Information Mutuelle	5
2.4.3	Processus suivi pour la sélection des features	5
2.4.4	Conclusion	6
<b>3</b>	<b>Modèles classiques et résultats</b>	<b>7</b>
3.1	Régression Logistique : Un Modèle Baseline	7
3.1.1	Principe et Fonctionnement	7
3.1.2	Pourquoi tester la régression logistique en reconnaissance des émotions ?	7
3.1.3	Expérimentation et Résultats	7
3.1.4	Conclusion	8
3.2	Machines à Vecteurs de Support (SVM)	8
3.2.1	Principe et Fonctionnement	8
3.2.1.1	Bases du SVM	8
3.2.1.2	Cas des Données Non Linéairement Séparables	8
3.2.1.3	Extension aux Problèmes Multi-Classes	9
3.2.1.4	Première Implémentation et Résultats	9
3.2.2	Optimisation des Hyperparamètres et des Features	10
3.2.2.1	Optimisation des Hyperparamètres par Grid-Search	10
3.2.2.2	Optimisation des Features avec l'Information Mutuelle	10
3.2.2.2.1	Optimisation naïve du nombre de MFCC	10
3.2.2.2.2	Sélection des Features basée sur l'Information Mutuelle	11
3.2.2.2.3	Conclusion intermédiaire	12
3.2.3	Résultats du SVM	12
3.2.3.1	Impact du seuil MI sur la précision	12
3.2.3.2	Performances finales du modèle SVM	13
3.2.3.3	Analyse des résultats	13
3.3	Random Forest	14
3.3.1	Avantages et Inconvénients du modèle	14
3.3.2	Fonctionnement des arbres de décisions	14
3.3.3	Fonctionnement des forêts aléatoires	14
3.3.4	Recherche des paramètres optimaux	15
3.3.5	Résultats finaux	17

<b>4</b>	<b>Réseaux avancés et résultats</b>	<b>18</b>
4.1	Cellules LSTM . . . . .	18
4.1.1	Introduction . . . . .	18
4.1.2	Principe de Fonctionnement des LSTM . . . . .	18
4.1.3	Résultats des LSTM . . . . .	20
4.1.3.1	Premiers résultats . . . . .	20
4.1.3.2	Résultats du modèle fine-tuné . . . . .	21
4.1.3.3	Résultats dans la littérature . . . . .	22
4.1.4	Conclusion . . . . .	22
<b>5</b>	<b>Conclusion et résultats</b>	<b>23</b>

# Chapitre 1

## Introduction

### 1.1 Contexte et objectif

### 1.2 Etat de l'art

#### 1.2.1 Présentation des bases de données

Nous nous sommes intéressés à quatre bases de données. La première d'entre elles est EMO-DB. Cette base de données présente certains avantages comme la qualité des enregistrements (la procédure décrite dans l'article de présentation de l'article [1]), ou une petite taille ce qui permet un entraînement plus rapide des modèles. La deuxième est CREMA D. Elle est presque douze fois plus grosse que EMO-DB [3]. Cela permet de tester des modèles nécessitant plus de données. Nous avons vu deux autres bases de données : RAVDESS et IMEOCAP. RAVDESS était beaucoup trop grosse et on ne pouvait donc pas entraîner des modèles dessus. Et nous n'avons pas accès à IMEOCAP.

Base de donnée	Taille	Autres	Retenue
EMO-DB (emotional database berlin)	40 MB	En Allemand, 10 Orateurs (5h, 5f), fréquence d'échantillonnage : 16kHz	<b>OUI</b>
CREMA D (Crowd Sourced Emotional Multimodal Actors Dataset)	473 MB	91 orateurs, 12 phrases différentes	<b>OUI</b>
RAVDESS	24.8 GB	24 orateurs, fréquence d'échantillonnage 48kHz	<b>NON</b>
IMEOCAP	?	Pas d'accès public	<b>NON</b>

TABLE 1.1 – Comparaison des bases de données

#### 1.2.2 Présentation de modèles sans deep learning

Le choix du modèle s'est fait sur une comparaison entre différents modèles. Nous avons retenus, pour les modèles sans deep learning, les forêts aléatoires et les machines à vecteurs de support. En effet, ces modèles ont en moyenne de très bonnes performances. [5].

Modèle	Inconvénients	Avantages	Performance
KNN (K nearest neighbours)	Classificateur linéaire... pour un problème non linéaire!	Facile à implémenter	Très mauvaise
Modèle de mélange gaussien	Difficulté Théorique Difficulté d'Implémentation	x	Bonne
Forêt aléatoire	x	x	Très bonne
Machines à vecteurs de support	x	Facile à implémenter	Très bonne
Modèles de Markov cachés	Apprentissage difficile	x	Moyenne

TABLE 1.2 – Comparaison des modèles d'apprentissage

Nous avons ensuite comparé nos résultats à ceux disponibles dans la littérature pour les mêmes modèles et bases de données [4].

Modèle\BDD	EMO-DB	CREMA D
SVM	<b>83.43</b> (min 77, max 93, N = 8)	<b>60.43</b> (min 56, max 63, N = 4)
Random Forest	<b>84.76</b> (min 80, max 89, N = 5)	<b>65.00</b> (min = 60, max = 70, N = 2)
HUMAIN	80.9	70

TABLE 1.3 – Comparaison des précisions des modèles sur différentes bases de données. N = nombre d'articles utilisés pour la statistique.

On remarque que les modèles ont relativement les mêmes performances, avec une légère différence sur CREMA D. On remarque aussi l'écart de performances entre CREMA D et EMO-DB. Cette différence se entre les deux bases de données se retrouve aussi dans l'oreille humaine. On retrouvera ces observations dans nos résultats dans la section "Conclusion et résultats".

### 1.2.3 Présentation de modèles avec deep learning

article

Modèle	Inconvénients	Avantages	Performance
MLP	Pas adapté aux signaux temporels	Facile à implémenter	Moyenne
CNN	Pas adapté aux signaux temporels	<b>Repère les motifs</b>	Bonne
LSTM	<b>Temps de calcul</b>	Modélise les relations temporelles	Très bonne
TRANSFORMER	<b>Difficulté théorique</b>	<b>Performances</b>	Très bonne

TABLE 1.4 – Comparaison des modèles d'apprentissage

Comme pour les modèles sans Deep Learning nous avons comparés les modèles avec Deep Learning pour voir lesquels accomplissaient le mieux la tâche étudiée. Sans surprise, les modèles les plus récents étaient très performants, nous avons donc choisis de les étudier dans la section 3. [2]

## Chapitre 2

# Préparation des données

### 2.1 Introduction

Le prétraitement des données est une étape essentielle en reconnaissance d'émotions à partir de la voix. Avant d'entraîner un modèle d'apprentissage automatique, il est crucial de transformer les signaux audio bruts en représentations exploitables, tout en minimisant le bruit et la redondance. Cette phase vise à extraire des caractéristiques pertinentes, également appelées features, à normaliser ces données et à sélectionner les meilleures features, afin améliorer la robustesse et la précision des modèles.

### 2.2 Caractéristiques Exploitées

Nous avons envisagé d'exploiter de nombreuses caractéristiques audios. Certaines ainsi que leur intérêt sont présentées ici.

- Les **MFCC (Coefficients Cepstraux en Fréquence Mel)** sont obtenus en appliquant plusieurs transformations successives au signal audio :

Le signal audio est segmenté en fenêtres, puis une transformée de Fourier est appliquée :

$$X(f) = \mathcal{F}\{x(t)\}$$

La fréquence  $f$  est transformée en échelle de Mel selon :

$$f_{\text{Mel}} = 2595 \log_{10} \left( 1 + \frac{f}{700} \right)$$

On applique un banc de filtres triangulaires pour obtenir les énergies des bandes de fréquences :

$$E_m = \sum_{k=0}^{N-1} |X(k)|^2 H_m(k)$$

où  $H_m(k)$  représente la réponse du  $m^{\text{ième}}$  filtre de Mel.

Enfin, on applique la transformée en cosinus discrète, qui permet de représenter un signal en termes de fonctions cosinus de différentes fréquences, pour obtenir les coefficients cepstraux :

$$C_n = \sum_{m=1}^M \log(E_m) \cos \left[ n \frac{\pi}{M} \left( m - \frac{1}{2} \right) \right]$$

où  $C_n$  est le  $n^{\text{ième}}$  coefficient MFCC.

- **Energie** : Sonie (volume sonore tel que perçu par l'humain) / Energie moyenne (root-mean-square).
- **Pitch & Jitter** : Fréquence fondamentale et ses variations à court terme. Cette grandeur est notamment calculable à partir de l'autocorrélation du signal. Trouver la fréquence fondamentale d'un signal revient à trouver la valeur de  $\tau$  pour laquelle la fonction suivante a un premier pic.

$$R(\tau) = \sum x(n) * x(n + \tau)$$

- **Formants** : Ils sont des pics de résonance dans le spectre de fréquences d'un son et sont généralement estimés à partir de la transformée de Fourier. Ils sont liés à la résonance du conduit vocal et jouent un rôle clé dans la perception des voyelles.

## 2.3 Une Première Solution avec *OpenSMILE*

Nous présentons dans cette partie une première solution envisagée pour l'extraction des données et leur préparation à la phase d'apprentissage. Les solutions envisagées plus tard dans le projet suivent le schéma de celle-ci, avec d'autres bibliothèques comme *librosa* et d'autres caractéristiques. La préparation des données se déroule comme suit.

Dans un premier temps, nous employons la bibliothèque *OpenSMILE* pour extraire les caractéristiques résumées dans le tableau suivant.

<b>Fréquence</b>	F0 (pitch), Jitter
<b>Amplitude</b>	Shimmer
<b>Rapport</b>	Harmonic-to-Noise Ratio (HNR)
<b>Énergie</b>	Loudness, Energy, Zero-Crossing Rate (ZCR)
<b>Coefficients cepstraux</b>	MFCC 1, MFCC 2, MFCC 3, MFCC 4
<b>Spectral</b>	Spectral Centroid, Spectral Flux, Spectral Entropy, Spectral Variance, Spectral Skewness, Spectral Kurtosis, Spectral Slope, Spectral Flatness, Spectral Sharpness, Spectral Harmonicity
<b>Formants</b>	Formants 1, Formants 2, Formants 3
<b>Largeur de bande des formants</b>	Formant Bandwidth 1, Formant Bandwidth 2, Formant Bandwidth 3

FIGURE 2.1 – Résumé des caractéristiques extraites par défaut avec *OpenSMILE*.

*OpenSMILE* prend en entrée chaque audio de notre base de données, le découpe en fenêtres de longueur réglable, puis extrait sur chaque fenêtre la série de caractéristiques présentées précédemment, et renvoie un tableau de dimensions *nombre de fenêtres*  $\times$  *nombre de caractéristiques*. Nous avons choisi une longueur de fenêtre de 40 ms avec un pas de 10 ms. Le pas de 10 ms permet d'avoir suffisamment de données et de bien suivre les évolutions rapides de la parole, tandis que la longueur de fenêtre fixée à 40 ms permet de une bonne résolution fréquentielle tout en permettant de considérer le signal quasi-stationnaire sur une fenêtre.

Dans un deuxième temps, nous appliquons aux données extraites une normalisation min-max entre 0 et 1. L'intérêt de la normalisation est multiple. D'une part, elle rend notre modèle plus robuste. En effet, si nous utilisons plusieurs bases de données pour entraîner un modèle d'apprentissage, il est tout à fait imaginable que les enregistrements d'une d'elles présentent un volume plus élevé que ceux d'une autre, simplement car elles ont été enregistrées différemment. La normalisation permet de ne plus avoir ce problème, qui n'est qu'un exemple. La normalisation évite d'autre part que certaines caractéristiques, ayant des valeurs très élevées par rapport aux autres, ne dominent l'apprentissage. Cela est important pour les modèles utilisant des réseaux de neurones, où des échelles de valeurs incohérentes peuvent ralentir la descente de gradient.

Dans un troisième temps, nous appliquons du padding à nos données. Le padding consiste à compléter les séquences de données afin qu'elles aient toutes la même longueur. Dans notre cas, chaque fichier audio est segmenté en fenêtres temporelles, et le nombre de fenêtres varie d'un fichier à l'autre en fonction de sa durée. Afin de pouvoir traiter nos données, certains modèles d'apprentissage nécessitent des entrées de taille fixe, nous ajoutons des valeurs de remplissage aux séquences plus courtes pour qu'elles atteignent une longueur maximale définie.

Le padding est généralement effectué en ajoutant une valeur spécifique aux caractéristiques des séquences incomplètes. Ici, les données ont été normalisées entre 0 et 1, la valeur choisie ne devait donc pas être comprise entre ces deux valeurs, et nous avons choisi  $-1$ . Lors de l'entraînement, l'algorithme d'apprentissage est conçu pour ignorer  $-1$  lors du calcul des poids et des mises à jour du modèle.

## 2.4 Optimisation du choix des features avec l'Information Mutuelle

### 2.4.1 Définition de l'Information Mutuelle

L'Information Mutuelle (MI) est une mesure de dépendance entre deux variables aléatoires. Elle quantifie la réduction de l'incertitude sur une variable donnée lorsqu'on connaît l'autre. Mathématiquement, elle est définie par :

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \quad (2.1)$$

où  $P(x, y)$  représente la probabilité conjointe des variables  $X$  et  $Y$ , et  $P(x)$ ,  $P(y)$  sont leurs distributions marginales.

L'Information Mutuelle permet d'identifier les features les plus pertinentes pour la classification des émotions en évaluant leur capacité à réduire l'incertitude sur les labels.

### 2.4.2 Défis liés à l'utilisation de l'Information Mutuelle

L'application de l'Information Mutuelle dans le choix des features pose plusieurs défis :

- **Lien MI/prédictibilité des labels** : Une valeur élevée de MI n'implique pas forcément une bonne séparation des classes. Il est donc nécessaire de valider empiriquement la pertinence des features sélectionnées.
- **Redondance entre features** : Certaines features ayant une forte Information Mutuelle avec les labels peuvent être corrélées entre elles, ce qui risque d'introduire de la redondance dans le modèle et d'augmenter la dimensionnalité sans réelle amélioration de la performance.

### 2.4.3 Processus suivi pour la sélection des features

Afin d'optimiser le choix des features tout en minimisant la redondance, nous avons suivi la méthodologie suivante :

1. **Extraction des features** : Nous avons extrait plusieurs features acoustiques utilisées couramment en reconnaissance d'émotions, notamment :
  - **MFCC (Mel-Frequency Cepstral Coefficients)** : Capturent la structure spectrale de la voix.
  - **Jitter** : Mesure de l'instabilité de la fréquence fondamentale.
  - **Shimmer** : Variation de l'amplitude du signal vocal.
2. **Calcul de l'Information Mutuelle** : Chaque feature a été évaluée selon son Information Mutuelle avec les labels des émotions afin de quantifier son utilité pour la classification.
3. **Sélection des features les plus pertinentes** : Seules les features ayant une Information Mutuelle au-dessus d'un seuil prédéfini ont été conservées, permettant de réduire le bruit et d'optimiser les performances du modèle.
4. **Validation des features sélectionnées** : Les performances du modèle ont été testées avec les features retenues à l'aide de classificateurs tels que le **SVM (Support Vector Machine)** et la **régression logistique** pour vérifier leur impact sur la séparation des classes.



#### 2.4.4 Conclusion

L'utilisation de l'Information Mutuelle nous a permis de mettre en place un outil efficace pour la sélection des features les plus discriminantes. Cette approche garantit que seules les informations les plus pertinentes sont conservées, facilitant ainsi l'apprentissage des modèles de classification tout en réduisant la complexité computationnelle. En combinant cette sélection avec des tests de validation, nous disposons désormais d'un pré-traitement robuste qui favorise une meilleure généralisation des modèles d'apprentissage automatique.

# Chapitre 3

## Modèles classiques et résultats

### 3.1 Régression Logistique : Un Modèle Baseline

Bien que la régression logistique ne soit pas un modèle de référence en reconnaissance des émotions, elle a été utilisée ici comme baseline afin d'évaluer l'impact des features avant d'appliquer des modèles plus avancés. Cette approche permet d'analyser la pertinence des caractéristiques extraites et de servir de point de comparaison pour les modèles plus complexes.

#### 3.1.1 Principe et Fonctionnement

La régression logistique est un modèle de classification linéaire utilisé principalement pour des tâches où l'on cherche à attribuer une probabilité à une classe donnée. Elle fonctionne en attribuant des coefficients aux différentes features pour générer une valeur :

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n \quad (3.1)$$

Cette valeur est ensuite transformée en probabilité à l'aide d'une fonction sigmoïde :

$$P(y = 1|X) = \frac{1}{1 + e^{-z}} \quad (3.2)$$

Dans un contexte de classification multi-classes, plusieurs techniques existent pour adapter la régression logistique, notamment l'approche One-vs-All (OvA) et Softmax Regression. L'approche OvA consiste à entraîner un modèle de classification binaire pour chaque classe en la comparant à toutes les autres, tandis que Softmax permet d'étendre directement la régression logistique à plusieurs classes simultanément en normalisant les scores.

#### 3.1.2 Pourquoi tester la régression logistique en reconnaissance des émotions ?

Même si la régression logistique n'est pas optimisée pour la classification de signaux audio complexes, elle présente certains intérêts :

- Elle constitue un bon point de comparaison avec des modèles plus complexes.
- Elle permet d'évaluer la qualité des features avant d'utiliser des modèles plus sophistiqués.
- Elle est rapide à entraîner et offre une interprétabilité facile des features utilisées.

Cependant, ses limites sont évidentes dans notre contexte :

- **Hypothèse de linéarité** : Les relations entre les features acoustiques et les émotions ne sont pas toujours linéaires.
- **Absence de prise en compte du contexte temporel** : Contrairement aux modèles comme les LSTM ou les CNN, la régression logistique traite chaque fenêtre d'analyse indépendamment.
- **Sensibilité au bruit** : Le signal vocal est riche en variations et bruit, ce qui peut affecter les performances d'un modèle aussi simple.

#### 3.1.3 Expérimentation et Résultats

Pour évaluer les performances de la régression logistique, nous avons testé ce modèle sur le dataset Emo-DB en utilisant 13 coefficients MFCC comme features. Les résultats montrent une précision de **68%**, ce qui confirme ses limitations mais reste un indicateur de la pertinence des features utilisées.

Les classes comme la colère et la tristesse sont relativement bien reconnues, tandis que d'autres comme l'ennui et l'anxiété montrent des performances plus faibles, ce qui suggère que des modèles plus complexes sont nécessaires pour capturer leurs nuances.

Classe	Précision	Recall	F1-score	Support
Colère	0.74	0.88	0.81	26
Anxiété	0.64	0.50	0.56	14
Ennui	0.50	0.31	0.38	16
Dégoût	0.67	0.67	0.67	9
Joie	0.69	0.64	0.67	14
Neutre	0.63	0.75	0.69	16
Tristesse	0.79	0.92	0.85	12

TABLE 3.1 – Résultats de la régression logistique sur Emo-DB

### 3.1.4 Conclusion

En conclusion, la régression logistique constitue une baseline rapide et interprétable, mais elle présente des limites évidentes pour la reconnaissance d'émotions. Son incapacité à modéliser des relations complexes et des dépendances temporelles limite son efficacité sur des signaux audio riches en information. Ces résultats justifient la transition vers des modèles plus sophistiqués comme les SVM, Random Forest, et réseaux neuronaux qui permettront d'améliorer la précision et la robustesse du système.

## 3.2 Machines à Vecteurs de Support (SVM)

### 3.2.1 Principe et Fonctionnement

#### 3.2.1.1 Bases du SVM

Les Machines à Vecteurs de Support (SVM) sont des modèles d'apprentissage supervisé particulièrement efficaces pour les problèmes de classification. Elles sont utilisées pour identifier un **hyperplan optimal** qui sépare les différentes classes de manière à maximiser la **marge** entre les points de données les plus proches, appelés **vecteurs de support**.

L'idée principale derrière un SVM est de trouver cet hyperplan qui maximise la distance entre les instances des deux classes les plus proches. Mathématiquement, un hyperplan dans un espace de dimension  $n$  est défini par l'équation :

$$w \cdot x + b = 0 \quad (3.3)$$

où :

- $w$  est le vecteur des poids associé aux features,
- $x$  représente les caractéristiques des données,
- $b$  est un biais qui ajuste la position de l'hyperplan.

L'optimisation d'un SVM repose sur la minimisation de la norme de  $w$ , tout en garantissant que toutes les observations sont correctement classées. Ce problème se formalise sous la forme suivante :

$$\min \frac{1}{2} \|w\|^2 \quad \text{sous contrainte} \quad y_i(w \cdot x_i + b) \geq 1, \quad \forall i \quad (3.4)$$

où  $y_i$  représente les labels des classes (+1 ou -1). L'objectif est donc de trouver  $w$  et  $b$  de manière à obtenir la plus grande séparation possible entre les classes.

#### 3.2.1.2 Cas des Données Non Linéairement Séparables

Dans la réalité, les données sont rarement parfaitement séparables par une frontière linéaire. Pour traiter ces cas, les SVM utilisent une technique appelée **trick du noyau** (kernel trick). L'idée est de transformer les données dans un espace de dimension plus élevée où elles deviennent séparables de manière linéaire.

Un **noyau** est une fonction qui permet de calculer directement le produit scalaire dans cet espace transformé, sans avoir à projeter explicitement les données. Les noyaux les plus couramment utilisés sont :

- **Noyau linéaire** : utilisé lorsque les classes sont bien séparables par une ligne droite dans l'espace des features.
- **Noyau polynomial** : projette les données dans un espace de dimension plus élevée en appliquant une transformation polynomiale.
- **Noyau RBF (Radial Basis Function)** : transforme les données dans un espace de dimension infinie, ce qui permet de modéliser des séparations complexes.

La figure suivante illustre la différence entre un SVM utilisant un noyau linéaire et un SVM utilisant un noyau RBF :

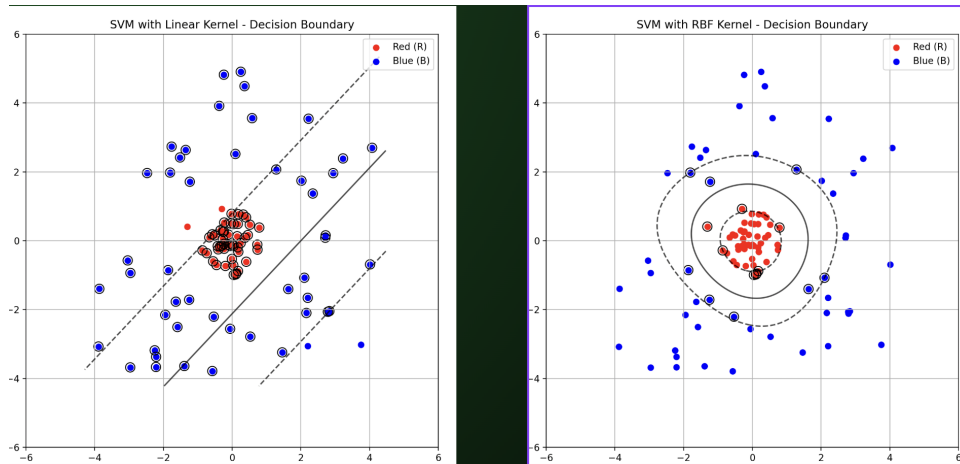


FIGURE 3.1 – Comparaison entre un SVM avec noyau linéaire (gauche) et un SVM avec noyau RBF (droite). On observe que le noyau RBF permet une séparation plus efficace des données non linéaires.

### 3.2.1.3 Extension aux Problèmes Multi-Classes

Les SVM ont été initialement conçus pour des problèmes de classification binaire (deux classes). Pour les adapter aux problèmes multi-classes, différentes stratégies sont utilisées :

- **One-vs-One (OvO)** : Un modèle SVM est entraîné pour chaque paire de classes, nécessitant ainsi  $\frac{n(n-1)}{2}$  SVM pour  $n$  classes. Lorsqu'un nouvel échantillon est classifié, un vote majoritaire est utilisé pour déterminer la classe finale.
- **One-vs-All (OvA)** : Un modèle SVM est entraîné pour distinguer chaque classe contre toutes les autres simultanément. Lors de la prédiction, la classe avec la plus grande confiance est choisie.

Dans notre projet, nous avons utilisé la méthode **One-vs-One** en raison de sa robustesse et de sa précision accrue sur des données fortement non linéaires.

### 3.2.1.4 Première Implémentation et Résultats

Pour valider notre approche initiale avec le SVM, nous avons réalisé un premier test sur le dataset **EMO-DB** en utilisant **13 coefficients MFCC** comme features. Cette première implémentation sert de point de référence avant d'optimiser les paramètres du modèle.

Les résultats obtenus indiquent une précision initiale de **71%**, comme présenté dans le tableau suivant :

Classe	Précision	Recall	F1-score	Support
Colère	0.76	0.85	0.80	26
Anxiété	0.58	0.50	0.54	14
Ennui	0.75	0.38	0.50	16
Dégoût	0.75	0.67	0.71	9
Joie	0.64	0.64	0.64	14
Neutre	0.68	0.94	0.79	16
Tristesse	0.79	0.92	0.85	12

TABLE 3.2 – Premiers résultats de la classification avec SVM sur EMO-DB.

Cette première expérimentation démontre que le SVM est capable de bien capturer certaines émotions (comme la colère et la tristesse) tout en rencontrant des difficultés pour d'autres (comme l'ennui et l'anxiété). Ces résultats nous motivent à explorer différentes stratégies d'optimisation pour améliorer la performance du modèle, sujet que nous aborderons dans la section suivante.

### 3.2.2 Optimisation des Hyperparamètres et des Features

Après avoir validé l'approche initiale avec un SVM et obtenu une précision de 71%, nous avons cherché à améliorer les performances en optimisant les hyperparamètres du modèle et en sélectionnant les features les plus pertinentes. Cette phase d'optimisation s'est déroulée en deux étapes principales :

- **Optimisation des hyperparamètres** via une recherche exhaustive (Grid-Search).
- **Sélection des features** les plus discriminantes à l'aide de l'Information Mutuelle (MI).

#### 3.2.2.1 Optimisation des Hyperparamètres par Grid-Search

Le SVM comporte plusieurs hyperparamètres influençant ses performances. Nous avons optimisé les trois paramètres suivants :

- **C** : coefficient de régularisation contrôlant le compromis entre maximisation de la marge et minimisation des erreurs de classification.
- **Type de noyau** : linéaire, polynomial ou RBF, influant sur la capacité du modèle à capturer des relations complexes.
- **Degré du noyau polynomial** : spécifique au noyau polynomial, permettant d'ajuster la complexité du modèle.

Nous avons utilisé une **Grid-Search** pour tester différentes combinaisons de ces hyperparamètres. La recherche a été effectuée sur les valeurs suivantes :

- *C* dans l'intervalle  $[0.1, 100]$
- *Kernel* : {linéaire, polynomial, RBF}
- *Degré* (pour le noyau polynomial) :  $[1, 7]$

Les performances des modèles ont été évaluées via une validation croisée sur plusieurs métriques :

- **Précision globale**.
- **Score F1** : équilibre entre précision et rappel.
- **Validation croisée** pour éviter le surapprentissage.

Le meilleur modèle trouvé correspond à un **SVM avec noyau linéaire** et un paramètre  $C = 0.1$ , suggérant que le dataset est globalement bien linéairement séparable et qu'une régularisation faible améliore la généralisation.

#### 3.2.2.2 Optimisation des Features avec l'Information Mutuelle

L'optimisation des features s'est déroulée en deux étapes complémentaires. La première approche est une recherche empirique naïve visant à identifier le nombre optimal de coefficients MFCC à utiliser. La seconde repose sur une méthode plus rigoureuse utilisant l'Information Mutuelle (MI) pour sélectionner les MFCC les plus informatifs.

**3.2.2.2.1 Optimisation naïve du nombre de MFCC** Dans cette première phase, nous avons fait varier le nombre de coefficients MFCC fournis en entrée du modèle et observé son impact sur la précision. L'objectif était d'identifier un point où ajouter plus de coefficients n'améliore plus significativement la performance.

La procédure adoptée était la suivante :

1. Faire varier le nombre de coefficients MFCC de manière croissante.
2. Entraîner le modèle pour chaque valeur et mesurer la précision obtenue.
3. Identifier le seuil où la performance se stabilise.

Les résultats de cette analyse sont présentés dans le graphique suivant :

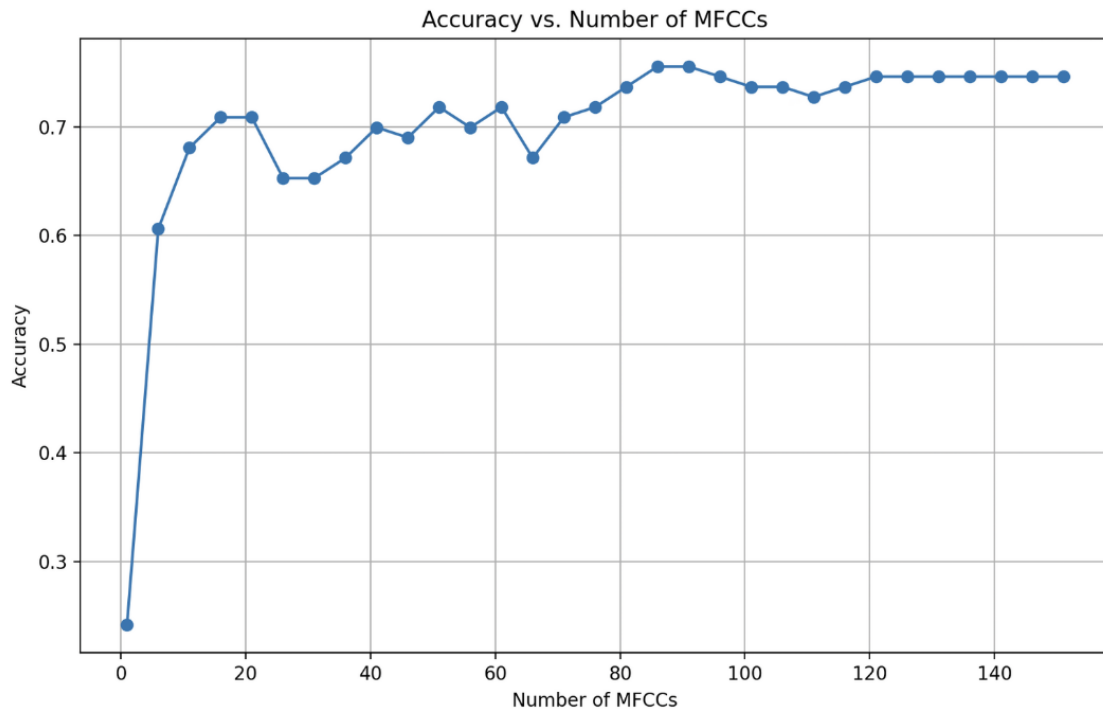


FIGURE 3.2 – Évolution de la précision en fonction du nombre de coefficients MFCC utilisés.

Nous avons constaté une augmentation rapide de la précision jusqu'à environ **86 coefficients MFCC**, après quoi elle tend à se stabiliser autour de **75.4%**. Ces résultats constituent une première indication du nombre de coefficients utiles, mais ils ne tiennent pas compte de la redondance possible entre les features.

**3.2.2.2.2 Sélection des Features basée sur l'Information Mutuelle** Pour affiner cette sélection, nous avons ensuite calculé les scores d'Information Mutuelle (MI) entre chaque coefficient MFCC et les labels de classification. L'Information Mutuelle permet de quantifier la dépendance entre une feature et la classe cible : un score élevé indique qu'un coefficient MFCC contient une forte information discriminante.

Le processus suivi était le suivant :

1. Calculer les scores MI pour chaque coefficient MFCC.
2. Définir un **seuil optimal** pour filtrer les features les moins informatives.
3. Sélectionner uniquement les MFCC dont le score dépasse ce seuil.

L'utilisation de cette approche permet de réduire davantage le nombre de coefficients utilisés tout en conservant ceux ayant un impact significatif sur la classification. Le graphique ci-dessous illustre la distribution des scores d'Information Mutuelle et la sélection effectuée :

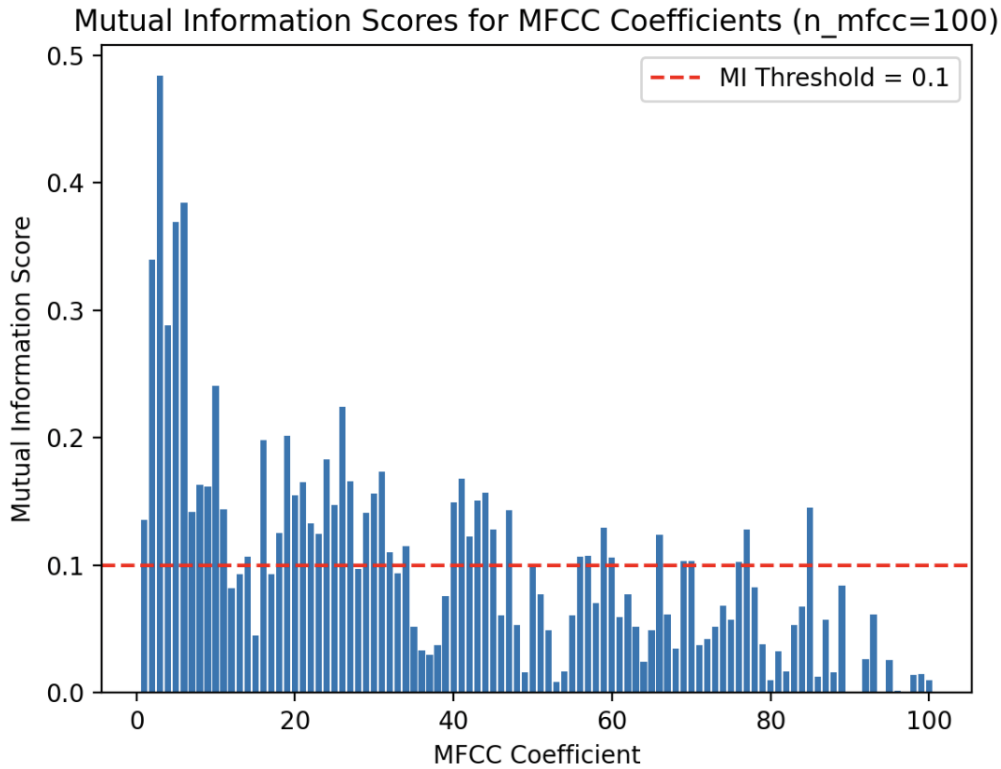


FIGURE 3.3 – Sélection des MFCC en fonction de leur score d'Information Mutuelle. Seuls les coefficients au-dessus du seuil sont conservés.

Cette méthode permet d'éliminer des features non informatives, améliorant ainsi la robustesse du modèle tout en réduisant sa complexité.

**3.2.2.2.3 Conclusion intermédiaire** Les résultats obtenus montrent que la première approche naïve permet d'identifier un nombre maximal de MFCC pertinents (**86**) et d'atteindre une précision de **75.4%**. Cependant, la sélection via l'Information Mutuelle apporte un cadre plus rigoureux pour ne conserver que les coefficients réellement discriminants.

Les performances finales après optimisation seront présentées et analysées dans la prochaine section dédiée aux résultats.

### 3.2.3 Résultats du SVM

Après l'optimisation des hyperparamètres et la sélection des features, nous avons évalué les performances finales du SVM afin de mesurer les gains obtenus par rapport à la première implémentation.

#### 3.2.3.1 Impact du seuil MI sur la précision

Pour affiner la sélection des coefficients MFCC, nous avons fait varier le seuil d'Information Mutuelle (MI) et mesuré son impact sur la précision du modèle. Un seuil trop bas inclut des features peu informatives et ajoute du bruit, tandis qu'un seuil trop élevé réduit drastiquement le nombre de features disponibles, diminuant ainsi la capacité du modèle à discriminer les classes.

Le graphique ci-dessous illustre cette variation :

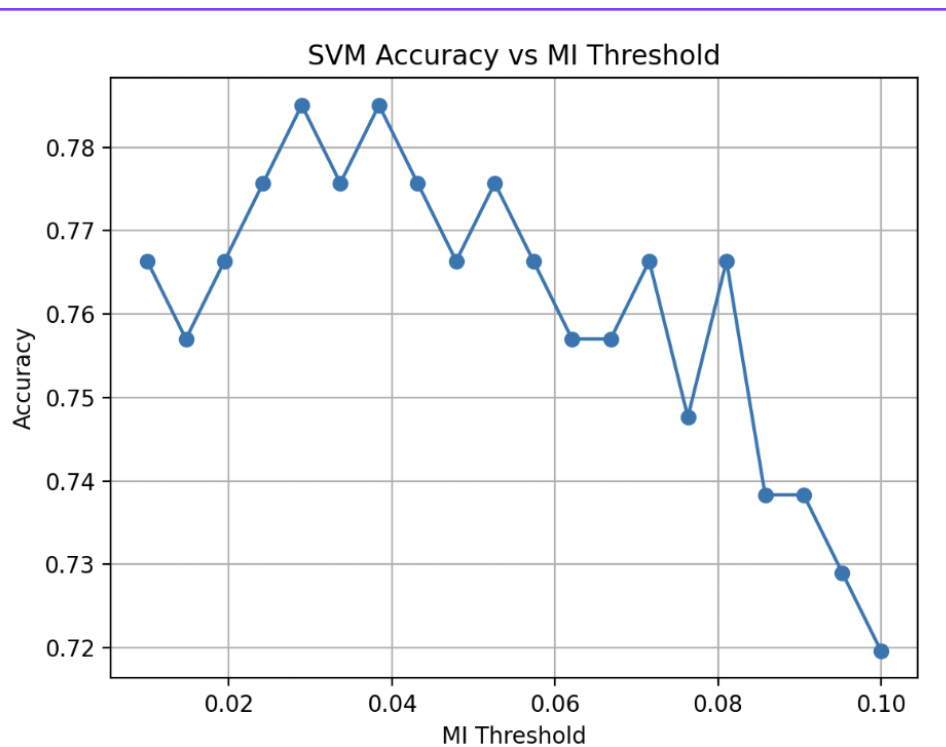


FIGURE 3.4 – Impact de la variation du seuil MI sur la précision du modèle.

Nous avons trouvé un seuil optimal autour de **0.04**, où la précision atteint un maximum avant de décroître lorsque trop de features sont éliminées.

### 3.2.3.2 Performances finales du modèle SVM

En appliquant les paramètres optimaux identifiés, nous avons obtenu une nette amélioration des performances du modèle. Le tableau suivant présente les résultats finaux de classification :

Classe	Précision	Recall	F1-score	Support
Colère	0.89	0.96	0.93	26
Anxiété	0.75	0.64	0.69	14
Ennui	0.68	0.81	0.74	16
Dégoût	0.67	0.71	0.69	9
Joie	0.79	0.79	0.79	14
Neutre	0.83	0.62	0.71	16
Tristesse	0.86	1.00	0.92	12

TABLE 3.3 – Résultats finaux de la classification avec SVM après optimisation.

En comparaison avec la première implémentation (précision de **71%**), le modèle optimisé atteint une précision de **80%**, avec une amélioration notable du **score F1 moyen**, qui passe de **69% à 78%**. Ces résultats confirment l'importance de l'optimisation des hyperparamètres et de la sélection des features dans l'amélioration de la performance.

### 3.2.3.3 Analyse des résultats

L'amélioration des performances est particulièrement marquée pour certaines classes, comme la colère et la tristesse, qui bénéficient d'un meilleur rappel et d'un score F1 plus élevé. En revanche, certaines classes restent plus difficiles à discriminer, notamment l'anxiété et le dégoût, dont les performances restent inférieures à celles des autres émotions. Les prochaines étapes consisteront à comparer ces résultats avec d'autres architectures plus complexes, notamment les réseaux de neurones, afin d'évaluer si des modèles plus avancés peuvent améliorer davantage la classification des émotions.



### 3.3 Random Forest

Le troisième modèle que nous avons essayé est le modèle des forêts aléatoires. Nous allons présenter son fonctionnement et les résultats obtenus avec ce modèle.

#### 3.3.1 Avantages et Inconvénients du modèle

AVANTAGES	INCONVÉNIENTS
<ul style="list-style-type: none"><li>— Permet de résoudre les problèmes de classification supervisée avec plusieurs catégories.</li><li>— Marche bien avec de gros datasets.</li><li>— Peut modéliser des relations non linéaires (cf exemple).</li><li>— Marchait bien dans la bibliographie.</li></ul>	<ul style="list-style-type: none"><li>— Non adapté aux données temporelles.</li></ul>

TABLE 3.4 – Avantages et inconvénients

#### 3.3.2 Fonctionnement des arbres de décisions

On peut représenter notre base de donnée traitée par un tableau de donnée  $x_i \in R^n$  pour  $i \in [1; l]$  et un vecteur label  $y \in R^l$ . On note  $Q$  l'arbre de décision que l'on veut construire. Les données au noeud  $m$  seront représentées par  $Q_m$ . Ces données contiennent  $n_m$  échantillons. Afin de décider de quoi sera composé le fils gauche et le fils droit du noeud  $Q_m$ , on choisit un couple  $\theta = (j, t_m)$  où  $j$  est une des features du tableau et  $t_m$  un seuil. On crée ainsi deux sous arbres :  $Q_m^{\text{left}}(\theta)$  et  $Q_m^{\text{right}}(\theta)$  subsets qui seront définis ainsi :

$$Q_m^{\text{left}}(\theta) = \{(x, y) \mid x_j \leq t_m\}$$

$$Q_m^{\text{right}}(\theta) = Q_m \setminus Q_m^{\text{left}}(\theta)$$

Etant donné que nous voulons faire l'arbre qui discrimine le mieux le label de la donnée, nous devons d'abord définir un critère d'"organisation de l'information". On introduit  $H()$  définit ainsi :

$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk})$$

où

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

Ici  $p_{mk}$  représente la proportion de donnée ayant le label  $k$  dans le noeud  $Q_m$ . La fonction  $H()$  quantifie l'entropie de la répartition des données dans les classes.

On introduit ensuite la fonction de perte :

$$G(Q_m, \theta) = \frac{n_m^{\text{left}}}{n_m} H(Q_m^{\text{left}}(\theta)) + \frac{n_m^{\text{right}}}{n_m} H(Q_m^{\text{right}}(\theta))$$

Ainsi, trouver les meilleurs paramètres pour scinder l'arbre revient à résoudre le problème :

$$\theta^* = \arg \min_{\theta} G(Q_m, \theta)$$

Puis on procède de manière récursive pour les nouveaux arbres créés par cette discrimination.

#### 3.3.3 Fonctionnement des forêts aléatoires

Les arbres aléatoires ont une forte tendance au sur-apprentissage. Pour pallier ce problème, on utilise en pratique les forêts aléatoires. Le principe des forêts aléatoires consiste à répéter l'algorithme des arbres aléatoires sur  $N$  échantillons de données extraits du tableau de données original, afin de générer des arbres différents. On retient ensuite le résultat majoritaire parmi ces arbres. La nature aléatoire de ce processus, appelé "Bagging" (Bootstrap Aggregating), permet de réduire le sur-apprentissage en diminuant la variance du modèle et en améliorant sa généralisation.

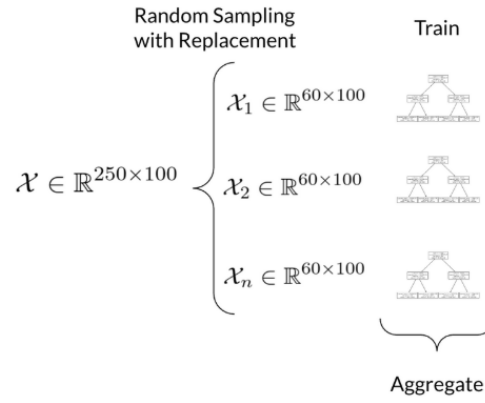


FIGURE 3.5 – Illustration du "Bagging"

### 3.3.4 Recherche des paramètres optimaux

Nous allons maintenant chercher les paramètres optimaux pour l'extraction des données et le modèle. Nous commençons par modifier la taille des arbres afin de déterminer si elle peut améliorer les résultats. Cette analyse sera effectuée sur les deux bases de données.

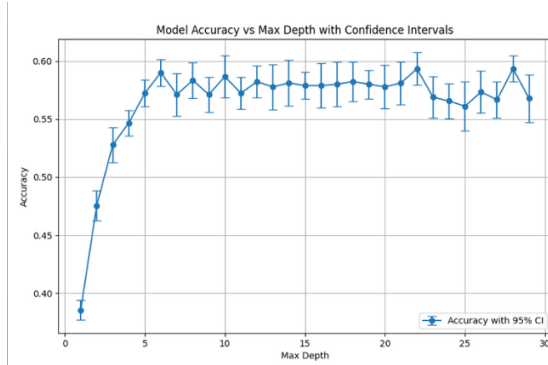


FIGURE 3.6 – Librairie opensmile pour l'extraction des données. Base de données utilisée : EMO DB

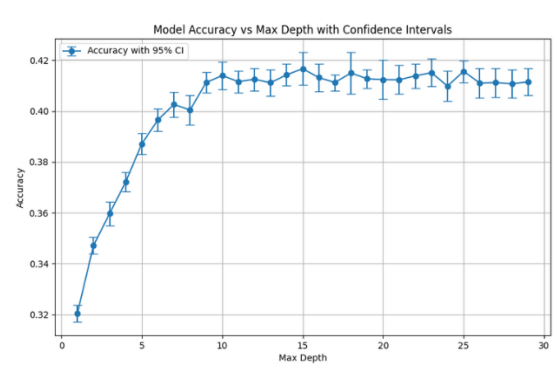


FIGURE 3.7 – Librairie opensmile pour l'extraction des données. Base de données utilisée : CREMA D

FIGURE 3.8 – Comparaison des précisions des modèles en fonction de la profondeur des arbres

Les code pour obtenir la figure 3.6 et la figure 3.7 portent respectivement le nom de "emo opensmile depth.py" et "crema opensmile depth.py". Ces fichiers se trouvent dans le dossier "models". Conformément à ce qui était attendu dans la littérature, le modèle performe mieux sur EMO-DB que sur CREMA-D, même si les résultats sont plus précis sur CREMA-D. On remarque aussi que, dans les deux cas, pour une profondeur supérieure à 10, la performance reste à peu près constante. Il n'est donc pas nécessaire d'avoir de très grands arbres pour obtenir de bons résultats.

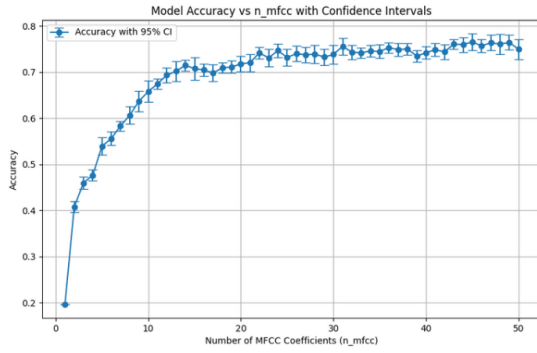


FIGURE 3.9 – Librairie librosa pour l'extraction des données. Base de données utilisée : EMO DB

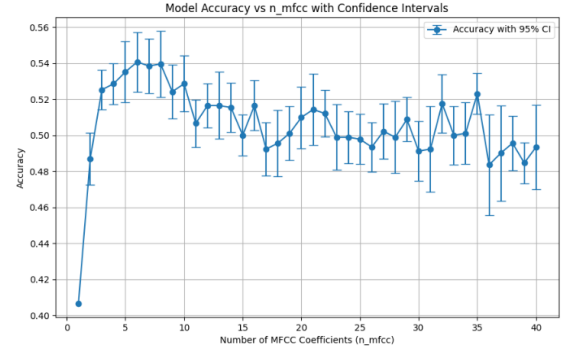


FIGURE 3.10 – Librairie librosa pour l'extraction des données. Base de données utilisée : CREMA D

FIGURE 3.11 – Comparaison des précisions des modèles en fonction du nombres de MFCC extraits

Les code pour obtenir la figure 3.9 et la figure 3.10 portent respectivement le nom de "emo librosa mfcc.py" et "crema librosa mfcc.py". Ces fichiers se trouvent dans le dossier "models". Encore une fois, le modèle est plus performant sur EMO-DB que sur CREMA-D, ce qui était attendu dans la littérature. De plus, on peut noter que les résultats sont beaucoup plus fiables sur EMO-DB que sur CREMA-D. On remarque que le modèle atteint ses meilleures performances pour des valeurs de MFCC supérieures à 30 sur la base de données EMO-DB. En revanche, pour la base de données CREMA-D, augmenter excessivement le nombre de MFCC réduit la précision. Maintenant que nous avons trouvé les paramètres optimaux, nous allons observer les résultats.

### 3.3.5 Résultats finaux

Maintenant que nous avons pu sélectionner la profondeur d'arbre et le nombre de MFCC optimaux, nous allons évaluer nos modèles.

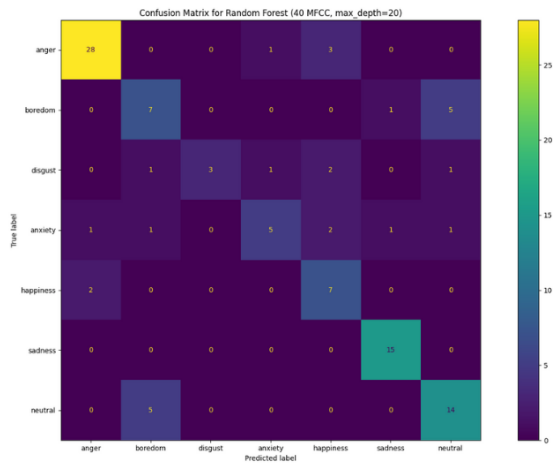


FIGURE 3.12 – Librairie librosa, base de donnée EMO DB, 40 MFCC, profondeur de l'arbre : 20

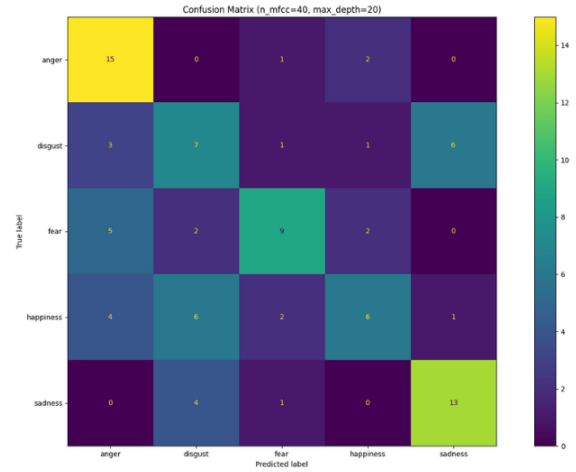


FIGURE 3.13 – Librairie librosa, base de donnée CREMA D , 40 MFCC, profondeur de l'arbre : 20

FIGURE 3.14 – Matrice de confusion des modèles

Les code pour obtenir la figure 3.12 et la figure 3.13 portent respectivement le nom de "conf matrix emo.py" et "conf matrix crema.py". Ces fichiers se trouvent dans le dossier "models". Pour la première matrice, les émotions sont très bien reconnues, hormis pour le dégoût et l'anxiété. On notera les très bons scores sur "anger" et "sadness", qui sont des émotions ayant été prononcées de façon très vive sur la base de données. Pour la deuxième matrice, les résultats sont plus mitigés et aucune émotion n'est très bien reconnue.

# Chapitre 4

## Réseaux avancés et résultats

### 4.1 Cellules LSTM

#### 4.1.1 Introduction

Les réseaux de neurones LSTM (Long Short-Term Memory) sont une variante des réseaux de neurones récurrents (RNN) qui sont particulièrement adaptés pour traiter des séquences temporelles. Dans le contexte de la reconnaissance d'émotions dans la voix, les LSTM présentent plusieurs avantages et inconvénients.

Avantages	Inconvénients
Réseau de neurones récurrent (RNN)	Mémoire à long terme parfois volatile
Comprennent les relations temporelles et non linéaires entre les features	Nécessite un grand dataset
Insensibles au décalage temporel	Effet boîte noire
Classification supervisée avec plusieurs catégories	Temps d'entraînement élevé

TABLE 4.1 – Avantages et inconvénients des LSTM

#### 4.1.2 Principe de Fonctionnement des LSTM

Les LSTM fonctionnent en utilisant des cellules de mémoire qui peuvent stocker, oublier ou mettre à jour des informations au fil du temps. Chaque cellule LSTM contient plusieurs composants clés : la cellule d'état, et trois portes (input gate, forget gate, output gate) qui régulent le flux d'informations.

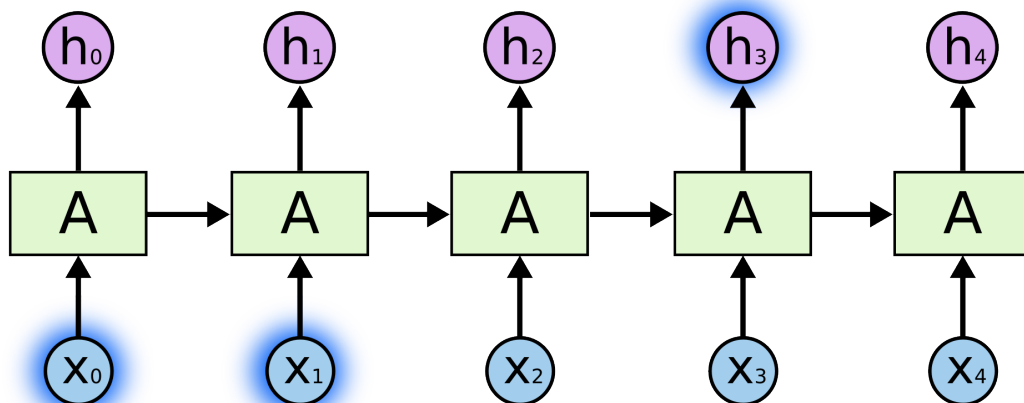


FIGURE 4.1 – Chaîne d'exécution d'un LSTM à une couche

**Cellule d'État :** La cellule d'état est le cœur du LSTM, elle permet de transmettre des informations sur de longues séquences. Elle est modifiée par les trois portes.

**Portes :** Les portes sont des mécanismes qui contrôlent le flux d'informations dans et hors de la cellule d'état. Elles utilisent des fonctions d'activation sigmoïde pour décider quelles informations doivent être conservées ou oubliées.

**Forget Gate :** La forget gate décide quelles informations de la cellule d'état doivent être oubliées. Elle prend en entrée l'état caché précédent  $h_{t-1}$  et l'entrée actuelle  $x_t$ , et produit un vecteur de valeurs entre 0 et 1, où 0 signifie "oublier complètement" et 1 signifie "conserver complètement".

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

**Input Gate** L'input gate décide quelles nouvelles informations doivent être ajoutées à la cellule d'état. Elle utilise deux étapes : d'abord, une couche sigmoïde décide quelles valeurs seront mises à jour, puis une couche tanh crée un vecteur de nouvelles valeurs candidates  $\tilde{C}_t$ .

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

La cellule d'état est ensuite mise à jour en utilisant les valeurs de la forget gate et de l'input gate.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

**Output Gate :** L'output gate décide quelles informations de la cellule d'état doivent être transmises à l'état caché. Elle utilise une couche sigmoïde pour décider quelles parties de la cellule d'état doivent être transmises, puis applique une fonction tanh à la cellule d'état et multiplie les deux résultats.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

La figure suivante illustre la chaîne d'exécution d'un LSTM à une couche et la représentation récurrente.

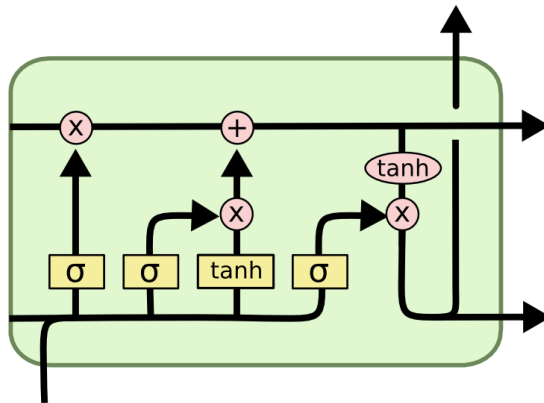


FIGURE 4.2 – L'intérieur d'une cellule LSTM

Toutefois, la nature séquentielle des LSTM impose de traiter différemment les données : Au lieu de récupérer l'ensemble des features sur tout un enregistrement et le définir en tant que vecteur feature, on donne séquentiellement au LSTM les  $i$ -èmes features (calculés sur la  $i$ -ème fenêtre). **L'output du modèle est donné seulement par la dernière forwardpropagation du modèle.** L'avantage principal est que le padding n'est plus nécessaire.

### 4.1.3 Résultats des LSTM

#### 4.1.3.1 Premiers résultats

Les résultats des LSTM sur les bases de données utilisées sont présentés dans les tableaux suivants. Il faut noter que le nombre total de paramètres dans ce modèle grandit rapidement avec le nombre de couches  $n_l$ , la taille de l'entrée  $n_i n$  et la dimension de l'information cachée  $n_c$  selon :

$$P = O(n_l n_c (n_c + n_{in}))$$

Pendant la phase d'entraînement, on a d'abord utilisé un modèle trop complexe pour le problème. Comme en témoignent les résultats ci-dessous, ce modèle a overfitté et avait du mal à généraliser.

	precision	recall	f1-score	support
anger	0.70	0.92	0.79	25
disgust	0.50	0.50	0.50	10
fear	0.38	0.43	0.40	14
happiness	0.40	0.31	0.35	13
neutral	0.57	0.53	0.55	15
sadness	1.0	0.57	0.73	14
accuracy			0.59	91
macro avg	0.59	0.54	0.55	91
weighted avg	0.61	0.59	0.59	91

TABLE 4.2 – Résultats des LSTM trop grands sur Emo-DB

Plusieurs points problématiques ont été identifiés lors de l'entraînement du modèle. Notamment, le nombre d'époques, le taux de dropout (0,2), la taille des batchs (32), le nombre de cellules LSTM (4), et la taille de la couche cachée (100) ont tous été jugés trop faibles. Ces paramètres insuffisants ont conduit à un modèle trop complexe, incapable d'apprendre correctement, comme l'indique la courbe de perte qui montre un manque de robustesse et un surajustement (overfitting).

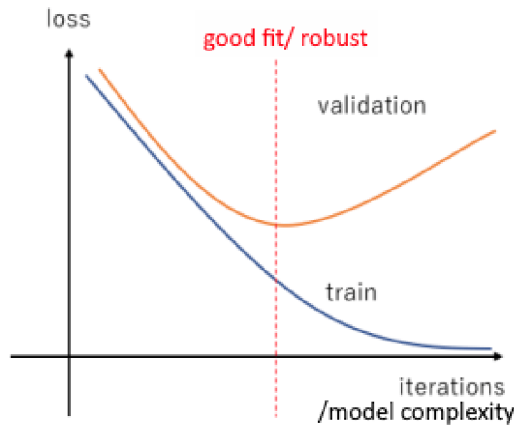


FIGURE 4.3 – Exemple de courbe d'overfitting

Le graphique exemple de la perte (loss) illustre clairement ce problème : la courbe de validation (en orange) diverge de la courbe d'entraînement (en bleu), ce qui est un signe typique d'overfitting. Cela signifie que le modèle mémorise les données d'entraînement au lieu de généraliser correctement aux nouvelles données. En d'autres termes, le modèle est trop complexe par rapport à la quantité et à la qualité des données disponibles, ce qui le rend sensible aux variations et aux bruits dans les données d'entraînement. Pour remédier à cela, il serait nécessaire d'ajuster les hyperparamètres, comme augmenter le taux de dropout, réduire la taille des batchs, ou simplifier l'architecture du modèle pour améliorer sa capacité de généralisation.

#### 4.1.3.2 Résultats du modèle fine-tuné

Pour pouvoir générer autant de tests on a utilisé le **Data Center d’Enseignement** de Metz.

	precision	recall	f1-score	support
anger	0.84	0.93	0.88	28
disgust	0.27	0.50	0.35	6
fear	1.00	0.50	0.67	12
happiness	0.60	0.64	0.62	14
neutral	1.00	0.83	0.91	18
sadness	0.92	0.92	0.92	13
accuracy			0.78	91
macro avg	0.77	0.72	0.73	91
weighted avg	0.83	0.78	0.79	91

TABLE 4.3 – Résultats des LSTM fine tunés sur Emo-DB

On constate que disgust a du mal à être détectée, et que globalement les résultats sont nettement meilleurs. Voyons ce qu’il en est pour CremaD :

	precision	recall	f1-score	support
anger	0.73	0.70	0.71	242
disgust	0.57	0.52	0.54	254
fear	0.48	0.58	0.52	264
happiness	0.51	0.54	0.52	236
neutral	0.63	0.66	0.65	212
sadness	0.59	0.49	0.53	281
accuracy			0.58	1489
macro avg	0.59	0.58	0.58	1489
weighted avg	0.58	0.58	0.58	1489

TABLE 4.4 – Résultats des LSTM fine tunés sur CremaD

Les résultats sont décevants. Puisque ce n’est pas un problème de robustesse de modèle (il fonctionne mieux sur un dataset plus petit) on conclut que c’est le dataset CremaD lui-même qui est problématique.

On combine maintenant CremaD et emoDB entier pour générer un nouveau dataset et voir si la langue parlée importe dans l’interprétation des émotions. Ce dernier nous a permis d’atteindre les résultats suivants :

	precision	recall	f1-score	support
anger	0.75	0.66	0.70	297
disgust	0.62	0.42	0.50	255
fear	0.47	0.54	0.50	278
happiness	0.51	0.60	0.55	259
neutral	0.57	0.63	0.60	229
sadness	0.56	0.59	0.57	248
accuracy			0.57	1566
macro avg	0.58	0.57	0.57	1566
weighted avg	0.58	0.57	0.57	1566

TABLE 4.5 – Résultats des LSTM fine tunés sur le dataset fusionné



Les résultats sont légèrement moins satisfaisants : la langue a effectivement une importance dans l'interprétation des émotions.

Au cours de l'entraînement, nous avons constaté que les meilleurs résultats étaient atteints pour EmoDB où l'on a retiré l'émotion disgust :

	precision	recall	f1-score	support
anger	0.89	0.93	0.91	27
fear	0.67	0.50	0.57	8
happiness	0.71	0.83	0.77	12
neutral	0.83	0.83	0.83	18
sadness	1.00	0.94	0.97	17
accuracy			0.85	82
macro avg	0.82	0.81	0.81	82
weighted avg	0.85	0.85	0.85	82

TABLE 4.6 – Résultats des LSTM fine tunés sur Emo-DB sans disgust

On atteint une précision moyenne pondérée de **85%** ce qui est **5%** moins bien que la littérature. On explique cette différence par le fait que la littérature utilise des LSTM couplés à d'autres modèles.

#### 4.1.3.3 Résultats dans la littérature

Par exemple, pour référence, Riccardo Cantini a utilisé un LSTM bidirectionnel avec attention, obtenant une précision de 90% pour la détection des émotions sur la base EmoDB, contre seulement 75% sans attention [6]. MeidanGR, quant à lui, a développé un système LSTM pour la reconnaissance des émotions en temps réel, atteignant une précision de 87% sur des fichiers audio issus du dataset Ravdess [8]. Enfin, Ege Kesim et al. ont proposé une approche multimodale combinant LSTM et Transformer, avec une précision de 69% pour la reconnaissance des émotions sur des données provenant de CREMA-D [7].

#### 4.1.4 Conclusion

Les LSTM montrent des performances intéressantes pour la reconnaissance d'émotions dans la voix, malgré le temps d'entraînement élevé. Les résultats obtenus sur les bases de données EmoDB et CremaD (qui présente cependant des moins bons résultats) démontrent leur efficacité, bien que certaines émotions comme la peur et le dégoût soient moins bien détectées.

## Chapitre 5

# Conclusion et résultats

Nous résumons ici les résultats obtenus pour les différents modèles et datasets :

Obtenu	EMO-DB	CREMA D
SVM	79	57
Random Forest	77	54
LSTM	85	57

TABLE 5.1 – Performances des modèles obtenus

Alors que dans la littérature on trouve par exemple :

Littérature	EMO-DB	CREMA D
SVM	83	60
Random Forest	84	65
LSTM	90	69

TABLE 5.2 – Performances des modèles de la littérature

En conclusion, nos modèles ont réussi à classer correctement les émotions avec une précision d'environ 80% pour EmoDB et de 55% pour CremaD. Ces résultats sont inférieurs à ceux rapportés dans la littérature, qui indique une précision proche de 90% pour EmoDB et de 69% pour CremaD. Ces différences peuvent être attribuées aux techniques supplémentaires que nous n'avons pas implémentées.

# Bibliographie

- [1] Burkhardt, f., paeschke, a., rolfes, m., sendlmeier, w.f., weiss, b. (2005) a database of german emotional speech. *proc. interspeech 2005*, 1517-1520, doi : 10.21437/interspeech.2005-446.
- [2] Deep learning techniques for speech emotion recognition, from databases to models babak joze abbaschian \* , daniel sierra-sosa and adel elmaghraby.
- [3] H. cao, d. g. cooper, m. k. keutmann, r. c. gur, a. nenkova and r. verma, "crema-d : Crowd-sourced emotional multimodal actors dataset," in *ieee transactions on affective computing*, vol. 5, no. 4, pp. 377-390, 1 oct.-dec. 2014.
- [4] Two-layer fuzzy multiple random forest for speech emotion recognition in human-robot interaction (1), t1 - improving speaker-dependency/independency of wavelet-based speech emotion recognition v1 - (2), s. yan, l. ye, s. han, t. han, y. li and e. alasaarela, "speech interactive emotion recognition system based on random forest," 2020 international wireless communications and mobile computing (iwcmc), limassol, cyprus, 2020 (3), crema-d : Crowd-sourced emotional multimodal actors dataset (4) speech emotion recognition using multimodal feature fusion with machine learning approach (5) an analysis of large speech models-based representations for speech emotion recognition (6) speech emotion recognition using a multi-time-scale approach to feature aggregation and an ensemble of svm classifiers (8).
- [5] Ververidis, d., kotropoulos, c. (2006). emotional speech recognition : Resources, features, and methods. *speech communication*, 48(9), 1162-1181.
- [6] Riccardo Cantini. Lstm bidirectionnel avec attention pour la détection des émotions. *EmoDB*, 2025. Précision de 90% avec attention, 75% sans attention.
- [7] Ege Kesim et al. Reconnaissance multimodale des émotions avec lstm et transformer. *CREMA-D*, 2025. Précision de 69% avec un modèle combiné LSTM et Transformer.
- [8] MeidanGR. Reconnaissance des émotions en temps réel avec lstm. *Ravdess*, 2025. Précision de 87% sur des fichiers audio.