Emotion Classification

MERJANI Aymane
MOUQED Mohamed Reda
EL BAJDALI Ahmed
SAISSI Badr
AJERAME Zakaria

March 2023

Supervisors :
Simon LEGLAIVE
Catherine SOLADIE

Introduction

La classification des émotions à l'aide d'enregistrements vocaux et de modèles d'apprentissage automatique, également connue sous le nom de Speech Emotion Recognition (SER), est une tâche essentielle dans les domaines du traitement du langage naturel (NLP) et de l'interaction homme-machine (HCI). La SER vise à développer des modèles d'apprentissage automatique capables d'identifier les émotions exprimées dans les enregistrements audio, tels que les voix humaines, et est cruciale pour diverses applications, notamment les assistants virtuels, le service clientèle et les diagnostics de santé mentale. La classification des émotions est souvent utilisée dans des domaines tels que la psychologie, la linguistique, la communication, la reconnaissance vocale et la robotique sociale.

Les modèles d'apprentissage automatique utilisés pour la classification des émotions à partir de données audio peuvent englober des techniques telles que les réseaux neuronaux, les arbres de décision, les machines à vecteurs de support (SVM) et les forêts aléatoires. Les caractéristiques acoustiques des signaux audio, telles que la hauteur, la durée, l'intensité et le débit de parole, sont souvent utilisées comme entrées dans ces modèles de classification.

La reconnaissance des émotions à partir de la parole peut s'avérer difficile en raison de la variabilité des émotions, des locuteurs et des langues, ainsi que du manque de données étiquetées et de la complexité de la tâche. La précision de la classification des émotions à partir de données audio varie en fonction de la qualité de l'enregistrement, du bruit de fond et de la variabilité individuelle de l'expression émotionnelle. Néanmoins, cette méthode offre des avantages uniques par rapport à d'autres techniques de détection des émotions, telles que la reconnaissance faciale, car elle peut être utilisée dans des contextes où la reconnaissance faciale n'est pas possible ou appropriée.

Ces dernières années, les approches d'apprentissage profond telles que les réseaux neuronaux convolutifs (CNN) et les réseaux neuronaux récurrents (RNN) ont donné des résultats prometteurs dans la classification des émotions à partir d'enregistrements vocaux. Les CNN sont particulièrement efficaces pour extraire les caractéristiques des spectrogrammes, qui sont des représentations graphiques du spectre de fréquence du signal audio au fil du temps. Les RNN, quant à eux, peuvent capturer efficacement la dynamique temporelle de l'expression émotionnelle dans la parole.

Table des matières

1 Etat de l'art

1.1 Features

1.1.1 Fréquence fondamentale f_0

```
f0 = librosa.yin(x, fmin=50, fmax=500)
# Calculée par l'algorithme de Yin
```

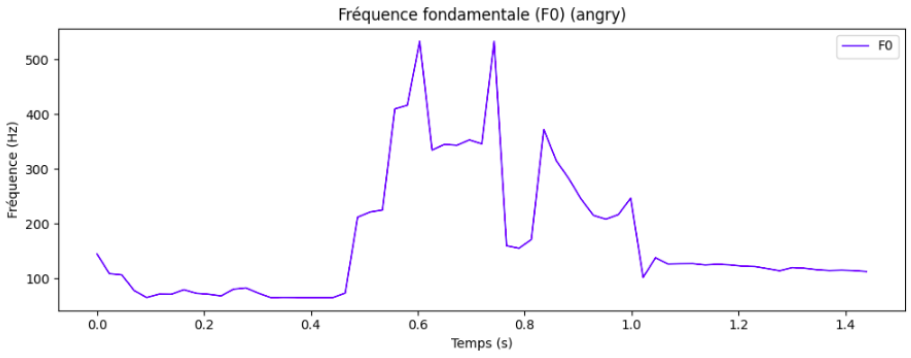
La fréquence fondamentale est la fréquence la plus basse des vibrations des cordes vocales lors de la production de la voix. La hauteur perçue d'un son est étroitement liée à la fréquence fondamentale, bien que la perception de la hauteur puisse également être influencée par d'autres facteurs, tels que les harmoniques présentes dans le signal.

La fréquence fondamentale est importante dans la classification des émotions pour plusieurs raisons :

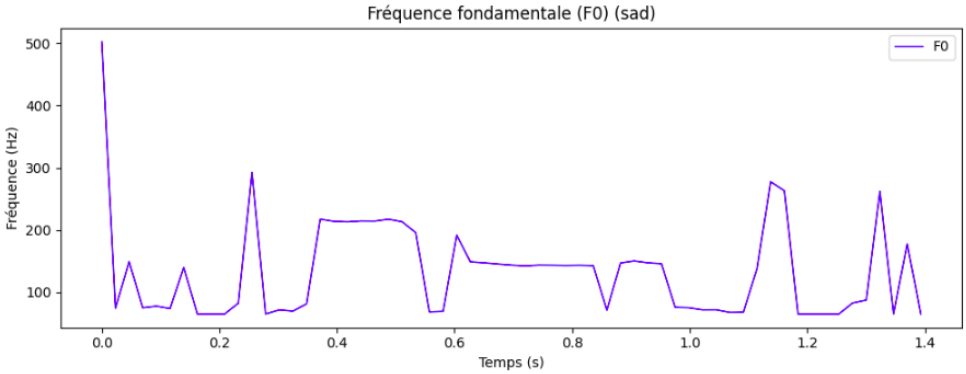
- Variation de la hauteur : Les émotions peuvent affecter la hauteur de la voix. Par exemple, la colère et la peur peuvent entraîner une augmentation de la hauteur, tandis que la tristesse et l'ennui peuvent provoquer une diminution de la hauteur.

- Variabilité de la hauteur : Certaines émotions, comme la surprise et l'excitation, peuvent provoquer une variabilité accrue de la hauteur au fil du temps, c'est-à-dire des changements rapides et fréquents de la hauteur.

- Stabilité de la hauteur : D'autres émotions, comme la tristesse, peuvent rendre la hauteur plus stable et moins variable au fil du temps.



On remarque dans cet exemple une augmentation rapide de la hauteur et un changement rapide des fréquences, c'est bien donc de la colère (angry). Les valeurs atteintes de la fréquence fondamentale ne sont pas assez réalistes mais ils sont comme même un peu élévées parce que c'était une voix de femme un peu gênée, qui est donc assez élévé en comparant avec l'homme.

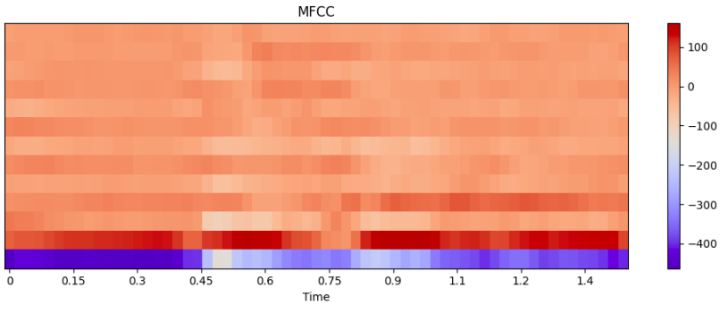


On remarque qu'il y a une diminution et une stabilité de la fréquence fondamentale en comparant avec la colère. C'est bien de la tristesse.

1.1.2 MFCC

```
mfccs = librosa.feature.mfcc(y=x, sr=sr, n_mfcc=13)
```

Mel-frequency cepstral coefficients : Ce sont des coefficients qui représentent les caractéristiques de fréquence d'un signal sonore. Ils sont calculés à partir de la transformée de Fourier à court terme (STFT) du signal sonore et représentent les fréquences qui sont les plus importantes pour la perception humaine du son.



L'axe horizontal représente le temps et l'axe vertical représente les coefficients MFCC. Chaque colonne de la matrice MFCC correspond à une trame (un court segment) du signal audio. L'intensité des couleurs à chaque point du graphique reflète la valeur d'un coefficient MFCC spécifique pour une trame spécifique. En général, les premiers coefficients MFCC représentent la forme spectrale globale du

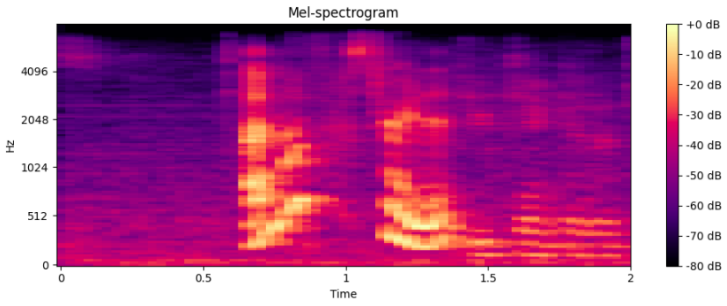
signal et sont plus importants pour les tâches de reconnaissance de la parole et des émotions. Les coefficients d'ordre supérieur capturent des informations spectrales plus détaillées.

1.1.3 Mel-spectrogram

```
mel_spectrogram = librosa.feature.melspectrogram(y=x, sr=sr)
```

Le Mel-spectrogramme est une représentation du spectre de fréquences d'un signal audio, où l'axe des fréquences est transformé en utilisant l'échelle de Mel. L'échelle de Mel est une échelle de fréquences perceptuelles qui imite la manière dont le système auditif humain perçoit les sons. Les fréquences sont converties en unités Mel, qui sont proportionnelles au logarithme des fréquences réelles. Cette conversion permet de donner plus de poids aux fréquences basses, qui sont généralement plus importantes pour la perception humaine.

Le Mel-spectrogramme est une représentation du spectre de fréquences basée sur l'échelle de Mel, tandis que les MFCC sont une représentation compacte et décorrélée obtenue à partir du Mel-spectrogramme.



1.1.4 Spectral centroid

Le centroïde spectral est une mesure qui décrit le "centre de gravité" du spectre de fréquences d'un signal audio. Il donne une idée de la répartition des fréquences dans le signal et peut être interprété comme une mesure de la "brillance" ou de la "luminosité" du son. Le centroïde spectral est calculé en prenant la moyenne pondérée des fréquences présentes dans le spectre, où les amplitudes des fréquences sont utilisées comme pondérations.

$$f_c = \frac{\sum S(k) f(k)}{\sum S(k)}$$

où $S(k)$ est l'amplitude spectrale à la fréquence de la bande k, $f(k)$ est la fréquence à la bande k.

Le centroïde spectral est important dans la classification des émotions pour plusieurs raisons :

- Caractéristiques du timbre : Le centroïde spectral est lié au timbre de la voix, qui est une caractéristique importante pour la perception des émotions. Les émotions peuvent affecter le timbre de la voix et donc modifier le centroïde spectral.

- Différences entre les émotions : Certaines émotions peuvent être associées à des centroïdes spectraux plus élevés, ce qui indique une voix plus brillante ou aiguë, tandis que d'autres émotions peuvent être associées à des centroïdes spectraux plus bas, ce qui indique une voix plus sombre ou grave.

- Complémentarité avec d'autres caractéristiques : Le centroïde spectral est complémentaire à d'autres caractéristiques audio.

1.1.5 Zero crossing rate (ZCR)

Le zero-crossing rate (taux de franchissement de zéro) est une mesure qui indique le nombre de fois qu'un signal audio change de signe, c'est-à-dire qu'il passe de positif à négatif ou vice-versa, au cours d'une période donnée (généralement une trame). Le zero-crossing rate est souvent utilisé comme une caractéristique pour décrire les propriétés temporelles d'un signal audio.

Un zero-crossing rate élevé peut indiquer la présence de bruit ou d'articulation rapide dans le signal vocal, tandis qu'un zero-crossing rate faible peut indiquer un son plus stable et moins bruyant. Ces différences peuvent être liées à des émotions spécifiques et aider à les distinguer.

1.2 Feature Importance

Dans le contexte de la reconnaissance des émotions à partir de données audio, l'extraction des caractéristiques joue un rôle essentiel dans la détermination de l'efficacité et des performances des modèles. La sélection des bonnes caractéristiques permet aux modèles de capturer les informations pertinentes des signaux audio, ce qui leur permet en fin de compte de reconnaître les émotions avec plus de précision. Dans cette section, nous discuterons de l'importance de l'extraction des caractéristiques et examinerons le graphique d'importance des caractéristiques obtenu à partir d'un classifieur Random Forest appliqué aux caractéristiques extraites.

Dans notre étude, nous avons utilisé un Classifieur Random Forest non seulement pour classer les émotions sur la base des caractéristiques extraites, mais aussi pour évaluer l'importance des caractéristiques. Le RandomForestClassifier est une méthode d'apprentissage d'ensemble qui construit plusieurs arbres de décision et combine leurs résultats pour obtenir une prédiction plus précise. Cette méthode permet également d'évaluer facilement l'importance des caractéristiques en analysant les informations obtenues à partir de chaque caractéristique lors de la construction des arbres.

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=1)
rf_classifier.fit(X_train, y_train)
```

Après avoir entraîné le classificateur, nous avons extrait les scores d'importance des caractéristiques et les avons classés par ordre décroissant.

```
feature_importances = pd.DataFrame(rf_classifier.feature_importances_,
index=X_test.columns, columns=['importance']).sort_values('importance', ascending=
False)
```

Nous avons ensuite visualisé les scores d'importance des caractéristiques à l'aide d'un bar plot

```
sns.barplot(x='importance', y=feature_importances.index, data=feature_importances)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Visualizing Important Features')
plt.tight_layout()
plt.show()
```

Le bar plot résultant permet de comprendre clairement quelles sont les caractéristiques les plus significatives pour faire des prédictions. En analysant le graphique d'importance des caractéristiques, nous pouvons identifier les caractéristiques les plus influentes dans l'ensemble de données, telles que la moyenne et l'écart-type de bandes de fréquences spécifiques dans le spectrogramme Mel. Ces informations peuvent être utiles pour affiner notre modèle et potentiellement améliorer ses performances.

En outre, la compréhension du contexte théorique de l'importance des caractéristiques peut aider à interpréter les résultats. Dans le contexte du RandomForestClassifier, l'importance des caractéristiques est calculée sur la base de la diminution de l'impureté, également connue sous le nom d'importance de Gini. La diminution de l'impureté est une mesure de la contribution d'une caractéristique spécifique à l'homogénéité du nœud résultant dans l'arbre de décision. Lors de la construction d'un arbre de décision, l'algorithme choisit la meilleure caractéristique à diviser en maximisant la diminution de l'impureté. En calculant la moyenne de la diminution de l'impureté pour chaque caractéristique dans tous les arbres de décision de la forêt aléatoire, nous obtenons une mesure de son importance pour la classification.

En résumé, l'importance des caractéristiques est un outil puissant pour comprendre la contribution des caractéristiques individuelles dans un modèle d'apprentissage automatique. En combinant le contexte théorique, la mise en œuvre du code et l'interprétation des résultats, nous pouvons acquérir une compréhension globale des caractéristiques les plus influentes dans notre ensemble de données, ce qui nous permettra de poursuivre le développement et l'affinement de notre modèle.

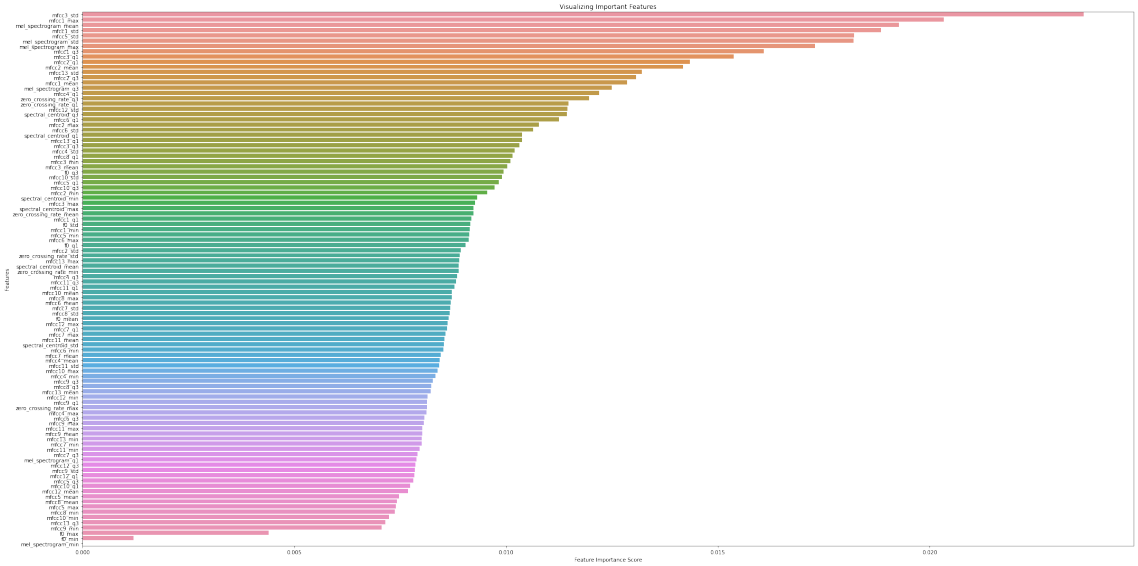


FIGURE 2 – Feature importance graph, généré avec un classifieur Random Forest

1.3 Modèles de Machine Learning

1.3.1 Random Forest

Le modèle de Machine Learning Random Forest est une technique d'apprentissage supervisé qui combine plusieurs arbres de décision pour créer un modèle plus robuste et précis. Les arbres de décision individuels sont construits à partir d'un échantillon aléatoire de données d'entraînement et de variables, et le résultat final est obtenu en agrégeant les prédictions de chaque arbre.

La méthode Random Forest peut être utilisée pour la classification et la régression. Pour la classification, le modèle prédit la classe la plus fréquente parmi les arbres individuels, et pour la régression, la prédiction finale est la moyenne des prédictions de chaque arbre. Cette technique permet de réduire le surapprentissage (overfitting) en limitant la profondeur de chaque arbre et en évitant la corrélation entre les prédictions.

Le modèle de Random Forest est très flexible et robuste, car il peut gérer des données manquantes, des valeurs aberrantes et des variables catégorielles sans nécessiter de prétraitement supplémentaire. Il peut également être utilisé pour extraire des informations sur l'importance des variables, ce qui permet d'identifier les caractéristiques les plus pertinentes pour la prédiction.

Hyperparamètres :

Le modèle Random Forest est un algorithme très flexible qui offre plusieurs paramètres qu'il est possible de varier pour l'optimiser selon les besoins. Voici les principaux paramètres qu'on peut ajuster dans le modèle de Random Forest :

— Le nombre d'arbres

— La profondeur maximale des arbres (`max_depth`) : elle détermine la profondeur maximale des arbres de décision dans le modèle. Une profondeur plus élevée peut améliorer la précision du modèle, mais peut également entraîner un surapprentissage.
— Le nombre minimum d'échantillons par feuille (`min_samples_leaf`) : il s'agit du nombre minimum d'échantillons requis pour créer une feuille de l'arbre. Une valeur plus élevée peut aider à éviter le surapprentissage, mais peut également réduire la précision du modèle.
— Le nombre minimum d'échantillons pour diviser un nœud (`min_samples_split`) : il s'agit du nombre minimum d'échantillons requis pour diviser un nœud en deux sous-groupes. Une valeur plus élevée peut aider à éviter le surapprentissage, mais peut également réduire la précision du modèle.
— Le critère de division (`criterion`) : il existe deux options pour le critère de division des nœuds : "gini" et "entropy". Gini est le critère de défaut et est généralement utilisé pour la classification, tandis que l'entropie est utilisée pour la régression.
— Le nombre maximum de variables (`max_features`) : il s'agit du nombre maximum de variables qui peuvent être utilisées pour créer chaque arbre de décision. Une valeur plus élevée peut améliorer la précision du modèle, mais peut également augmenter le temps de calcul.
— Le nombre aléatoire (`random_state`) : il s'agit d'une valeur aléatoire qui permet de reproduire les mêmes résultats à chaqexécution. Cela peut être utile pour la reproductibilité du modèle.

1.3.2 CNN

Dans la classification des émotions en machine learning, les Convolutional Neural Networks (CNN) peuvent être utilisés pour extraire des caractéristiques à partir de données d'entrée, telles que des images ou des sons.

Le principe de fonctionnement de CNN dans la classification des émotions consiste à appliquer plusieurs couches de convolution et de sous-échantillonnage pour extraire des motifs à partir des données d'entrée. Les couches de convolution utilisent des filtres pour détecter des motifs spécifiques dans les données d'entrée, tandis que les couches de sous-échantillonnage réduisent la taille des données en conservant uniquement les informations les plus importantes.

Une fois que les caractéristiques ont été extraites par les couches de convolution et de sous-échantillonnage, elles sont flattées en un vecteur et passées à travers une ou plusieurs couches entièrement connectées qui effectuent la classification des émotions.

L'entraînement du modèle de CNN dans la classification des émotions implique l'optimisation des poids et des biais des différentes couches du réseau pour minimiser une fonction de coût, telle que l'erreur quadratique moyenne ou la perte d'entropie croisée.

En résumé, le principe de fonctionnement de CNN dans la classification des émotions consiste à extraire des caractéristiques à partir des données d'entrée en utilisant des couches de convolution et de sous-échantillonnage, puis à effectuer la classification des émotions en utilisant des couches entièrement connectées. L'entraînement du modèle de CNN implique l'optimisation des poids et des biais des différentes couches pour minimiser une fonction de coût.

1.3.3 XGBoost

XGBoost, abréviation de eXtreme Gradient Boosting, est une puissante bibliothèque open-source développée pour implémenter des algorithmes de gradient boosting. Il est conçu pour être hautement efficace et scalable, fournissant une implémentation optimisée et parallélisée de l'algorithme Gradient Boosted Decision Trees (GBDT). XGBoost est largement utilisé en apprentissage automatique et en compétitions de science des données en raison de sa performance supérieure en termes de vitesse et de précision.

Gradient Boosting Le gradient boosting est une technique d'apprentissage ensembliste qui combine les prédictions de plusieurs apprenants faibles, généralement des arbres de décision, pour créer un apprenant fort. Il fonctionne en ajoutant itérativement de nouveaux modèles à l'ensemble, chaque nouveau modèle tentant de corriger les erreurs commises par le précédent. L'idée centrale derrière le gradient boosting est de minimiser une fonction de perte donnée en optimisant les prédictions du modèle à l'aide de la descente de gradient.

Caractéristiques de XGBoost

— **Régularisation :** XGBoost intègre à la fois la régularisation L1 (Lasso) et L2 (Ridge), qui aident à éviter le surapprentissage en pénalisant les modèles complexes.
— **Sensibilité à la parcimonie :** XGBoost peut gérer les données clairsemées et les valeurs manquantes de manière efficace, le rendant adapté à un large éventail d'applications.
— **Parallélisation :** XGBoost peut profiter de plusieurs cœurs de CPU, ce qui permet des temps d'entraînement plus rapides.
— **Validation croisée :** La bibliothèque offre une prise en charge intégrée de la validation croisée, simplifiant la sélection du modèle et l'évaluation des performances.

Algorithme XGBoost utilise l'algorithme des arbres boostés par gradient. À chaque itération, un nouvel arbre est construit qui vise à minimiser les erreurs résiduelles de l'ensemble existant. Les erreurs résiduelles sont calculées comme les gradients de la fonction de perte par rapport aux prédictions actuelles. Le nouvel arbre est ensuite ajouté à l'ensemble avec un facteur de pondération, appelé taux d'apprentissage, qui contrôle la contribution du nouvel arbre aux prédictions globales.

1.3.4 Hyperparamètres

Quelques hyperparamètres clés de XGBoost incluent :
— **n_estimators :** Le nombre d'arbres dans l'ensemble.
— **max_depth :** La profondeur maximale de chaque arbre.
— **learning_rate :** La taille de pas utilisée dans la descente de gradient.
— **subsample :** La fraction des données d'entraînement utilisée pour construire chaque arbre.

Conclusion :

XGBoost est une bibliothèque de gradient boosting efficace, scalable et polyvalente qui a gagné une popularité significative en raison de sa performance exceptionnelle dans diverses tâches d'apprentissage automatique. Sa capacité à gérer divers types de données, à incorporer des techniques de régularisation et à exploiter les capacités de traitement parallèle en font un outil incontournable pour les scientifiques des données et les praticiens de l'apprentissage automatique.

1.4 Indicateurs de performence

Il existe plusieurs indicateurs que l'on peut utiliser pour évaluer la performance d'un modèle de machine learning. Les indicateurs que l'on choisit dépendent du type de problème que l'on essaye de résoudre, de la nature des données et du modèle que l'on utilise. Voici quelques-uns des indicateurs les plus couramment utilisés :

1. Précision : La précision mesure la proportion de prédictions correctes par

rapport à l'ensemble des prédictions. C'est un indicateur couramment utilisé pour évaluer les modèles de classification.

2. Rappel : Le rappel mesure la proportion d'exemples positifs correctement identifiés par le modèle par rapport à l'ensemble des exemples positifs. C'est également un indicateur couramment utilisé pour évaluer les modèles de classification.
3. F-mesure : La F-mesure est une mesure qui combine la précision et le rappel en une seule valeur qui prend en compte les deux indicateurs. C'est un indicateur utile pour évaluer les modèles de classification lorsque l'on veut donner une importance égale à la précision et au rappel.
4. Erreur quadratique moyenne (EQM) : L'EQM mesure la moyenne des carrés des écarts entre les valeurs prédites et les valeurs réelles. C'est un indicateur couramment utilisé pour évaluer les modèles de régression.
5. Coefficient de détermination (R^2) : Le coefficient de détermination mesure la proportion de la variance de la variable cible qui est expliquée par le modèle. C'est un indicateur couramment utilisé pour évaluer les modèles de régression.
6. AUC (aire sous la courbe ROC) : L'aire sous la courbe ROC mesure la capacité d'un modèle de classification à faire la distinction entre les classes positives et négatives. C'est un indicateur couramment utilisé pour évaluer les modèles de classification binaire.
7. Log loss : La log loss mesure la performance d'un modèle de classification en prenant en compte à la fois la précision et la confiance du modèle dans ses prédictions. C'est un indicateur couramment utilisé pour évaluer les modèles de classification binaire.

Il est important de noter que ces indicateurs ne doivent pas être utilisés isolément pour évaluer un modèle de machine learning. Ils doivent être utilisés conjointement pour obtenir une évaluation complète de la performance du modèle. De plus, il est important de choisir les indicateurs en fonction des objectifs du projet et des données que l'on utilise.

2 Méthodologie

2.1 Objectifs

L'objectif principal de ce projet est de développer un modèle de reconnaissance des émotions par la voix (SER) capable de reconnaître avec précision les émotions à partir de la parole. Plus précisément, nous visons à atteindre les objectifs suivants :

- Pre-process les signaux vocaux et extraire les caractéristiques pertinentes pour représenter le contenu émotionnel des signaux vocaux.

- Développer un modèle de classification capable de classer avec précision les signaux vocaux dans les catégories d'émotions.

- Évaluer les performances du modèle proposé et les comparer à d'autres modèles de reconnaissance des émotions.

2.2 Base de données

2.2.1 IEMOCAP

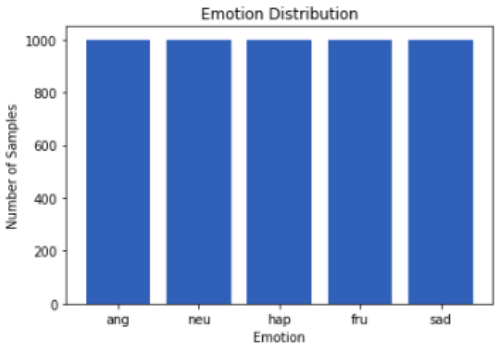
La base de données IEMOCAP est un ensemble de données pour la recherche en reconnaissance des émotions dans la parole, contenant une grande collection d'enregistrements audiovisuels d'interactions humaines, où les intervenants expriment une gamme d'émotions. La base de données comprend plus de 12 heures de données audiovisuelles, incluant des vidéos, des enregistrements vocaux, la capture de mouvements du visage, des transcriptions textuelles étiquetées avec le modèle VAD (Valence-Arousal-Dominance) et leur catégorie d'émotion correspondante (frustrated, neutral, angry, sad, excited, happy, surprised, fear, disgusted ...). Pour notre projet, nous utiliserons uniquement les enregistrements audio pour nos données pour classifier les émotions.

2.2.2 Pre-processing


```
xxx    2507
fru    1849
neu    1708
ang    1103
sad    1084
exc    1041
hap     595
sur     107
fea      40
oth       3
dis       2
Name: emotion, dtype: int64
```

Tout d'abord, on commence par importer des données et en faire une dataframe pandas, afin de pouvoir nous en servir pour extraire les features pour l'entraînement de notre modèle. Nous nous retrouvons donc avec plus de 10 000 données, avec plusieurs émotions : "fru" pour frustration, "neu" pour neutre, "ang" pour colère, "sad" pour tristesse, "exc" pour enthousiasme, "hap" pour la joie, "sur" pour la surprise, 'fea" pour la peur, "dis" pour le dégoût, et enfin "oth" pour autre et "xxx" lorsque c'est indéterminé.

Pour ces données, nous décidons de visualiser la distribution des émotions, afin de voir si nous avons une distribution uniforme. Nous remarquons donc que ce n'est pas le cas, et décidons donc de ne garder que les 5 premières émotions évoquées plus tôt en fusionnant "hap" avec "exc" . Nous nous retrouvons avec une nouvelle table de données contenant cette fois-ci plus de 5000 données.



2.2.3 Création des nouvelles bases de données

Le principal problème rencontré dans la reconnaissance des émotions dans la parole à partir de la base de données IEMOCAP est le grand nombre de caractéristiques extraites des signaux vocaux. Le défi est de sélectionner les features les plus pertinentes qui peuvent discriminer efficacement les différentes émotions. Toutes les features appartiennnent à la bibliothèque "librosa"

On a décidé de créer deux datasets.

- Une contenant : l'émotion de l'audio, sa session, sa moyenne, écart-type, 1er et 3e quartile, minimum, maximum de chaque feature : f0, 13 coefficients MFCC, coefficients du spectrogramme de Mel et ZCR.

- L'autre contenant les données brutes des Mel-spectrogram pour le CNN (convolutional neural network).

2.3 Implémentation des modèles

2.3.1 Random Forest

Le code commence par diviser les données en deux ensembles, un ensemble d'entraînement `train_data` et un ensemble de test `test_data`. Les ensembles d'entraînement et de test sont utilisés pour entraîner et tester le modèle de classification.

```
X_train = train_data.drop("emotion", axis=1)
y_train = train_data["emotion"]

X_test = test_data.drop("emotion", axis=1)
y_test = test_data["emotion"]
```

Ensuite, un objet de classe `RandomForestClassifier` est initialisé avec 100 arbres de décision `n_estimators` = 100 et une graine aléatoire de 42 `random_state` = 42. Le modèle est ajusté aux données d'entraînement en utilisant la méthode `fit`(). Une fois que le modèle a été ajusté, il est utilisé pour faire des prédictions sur l'ensemble de test avec la méthode `predict`(). Les prédictions sont stockées dans `y_pred`. Le code affiche ensuite un rapport de classification pour évaluer les performances du modèle. Il utilise la fonction `classification_report`() de scikit-learn pour afficher les scores de précision, de rappel et de F1 pour chaque classe, ainsi que la précision globale et le rappel du modèle.

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)

y_pred = rf_classifier.predict(X_test)

print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Le code calcule également une matrice de confusion pour évaluer la performance du modèle en détail. La matrice de confusion est tracée à l'aide de la bibliothèque `seaborn`, qui crée une carte de chaleur annotée de la matrice de confusion.

```
cm = confusion_matrix(y_test, y_pred, labels=emotion_classes)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=emotion_classes, yti
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

Enfin, le code affiche un graphique de la fonctionnalité d'importance pour visualiser les fonctionnalités les plus importantes utilisées par le modèle pour effectuer des prédictions. Le graphique est créé à l'aide de la bibliothèque `seaborn` pour tracer un graphique à barres des scores d'importance des différentes fonctionnalités.

```

Feature importance
feature_importances = pd.DataFrame(rf_classifier.feature_importances_, index=X_test
print("Feature Importances:")
print(feature_importances)

plt.figure(figsize=(12, 6))
sns.barplot(x='importance', y=feature_importances.index, data=feature_importances)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Visualizing Important Features')
plt.tight_layout()
plt.show()
```

2.3.2 CNN

Ce code est destiné à entraîner un modèle de réseau de neurones convolutifs (CNN) pour la classification des émotions à partir de spectrogrammes.

La première partie du code consiste à prétraiter les données en utilisant la bibliothèque scikit-learn. Les labels des émotions sont encodés à l'aide de la classe LabelEncoder et transformés en variables catégorielles à l'aide de la fonction `to_categorical`(). Les spectrogrammes sont normalisés en divisant par 255 et empilés dans un tableau numpy.

La deuxième partie du code construit le modèle CNN en utilisant la bibliothèque Keras. Le modèle est composé de couches de convolution 2D, de pooling maximal, d'une couche de sortie dense et d'une couche de sortie softmax. Le modèle est compilé

en utilisant l'optimiseur Adam, la fonction de perte `categorical_crossentropy` et la métrique de précision (accuracy).

La troisième partie du code utilise la bibliothèque Keras pour générer des données d'entraînement augmentées à l'aide de la classe ImageDataGenerator(). Les poids de classe sont calculés à l'aide de la fonction `compute_sample_weight`() pour tenir compte de la distribution inégale des classes dans les données.

La fonction model.fit() est utilisée pour entraîner le modèle. Les données d'entraînement sont fournies à la fonction flow() de l'objet ImageDataGenerator(), tandis que les données de test sont fournies directement. Les poids de classe sont passés à la fonction pour tenir compte de la distribution inégale des classes. Le nombre d'époques et la taille de lot (batch size) sont également spécifiés. Les performances du modèle sont enregistrées à chaque époque dans un objet history.

```
1
2  emotion_labels = df_spectrograms["emotion"].values
3  label_encoder = LabelEncoder()
4  emotion_labels_encoded = label_encoder.fit_transform(emotion_labels)
5  emotion_labels_one_hot = to_categorical(emotion_labels_encoded)
6
7  spectrograms = np.stack(df_spectrograms["spectrogram"].values) / 255.0
8  X_train, X_test, y_train, y_test = train_test_split(spectrograms, emotion_labels_on
9
10 X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1)
11 X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)
12
13 input_shape = (X_train.shape[1], X_train.shape[2], 1)
14 num_classes = 5
15
16 model = Sequential([
17     Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape),
18     MaxPooling2D(pool_size=(2, 2)),
19     Conv2D(64, kernel_size=(3, 3), activation='relu'),
20     MaxPooling2D(pool_size=(2, 2)),
21     Conv2D(128, kernel_size=(3, 3), activation='relu'),
22     MaxPooling2D(pool_size=(2, 2)),
23     Flatten(),
24     Dense(256, activation='relu'),
25     Dropout(0.5),
26     Dense(num_classes, activation='softmax')
27 ])
28
29 model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['
30 model.summary()
```

```
31
32 train_datagen = ImageDataGenerator(
33     rotation_range=15,
34     width_shift_range=0.1,
35     height_shift_range=0.1,
36     horizontal_flip=True
37 )
38
39 train_datagen.fit(X_train)
40
41 class_weights = compute_sample_weight(class_weight='balanced', y=emotion_labels)
42 class_weights = {i: weight for i, weight in enumerate(class_weights)}
43
44 batch_size = 32
45 epochs = 25
46
47 history = model.fit(train_datagen.flow(X_train, y_train, batch_size=batch_size),
48                     validation_data=(X_test, y_test),
49                     class_weight=class_weights,
50                     steps_per_epoch=len(X_train) // batch_size,
51                     epochs=epochs)
52
```

2.3.3 XGBoost

Dans cette section, nous discuterons de l'utilisation du classifieur XGBoost, un algorithme de Gradient Boosting puissant et largement utilisé, pour effectuer la reconnaissance des émotions à partir des caractéristiques données. Le choix de XGBoost est motivé par ses bonnes performances dans diverses tâches de classification et par sa capacité à traiter efficacement des ensembles de données vastes et complexes.

Tout d'abord, nous devons coder les étiquettes d'émotion en valeurs numériques, car le classificateur XGBoost a besoin d'entrées numériques. Pour ce faire, nous utilisons le LabelEncoder de scikit-learn :

```
1 label_encoder = LabelEncoder()
2 y_train_encoded = label_encoder.fit_transform(y_train)
3 y_test_encoded = label_encoder.transform(y_test)
```

Ensuite, nous initialisons et entraînons le classificateur XGBoost avec les paramètres spécifiés. Nous définissons `use_label_encoder=False` pour éviter d'utiliser l'encodeur d'étiquettes déprécié, `eval_metric='mlogloss'` pour utiliser la perte logarithmique multi-classe comme métrique d'évaluation.

```
xgb_classifier = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_classifier.fit(X_train_scaled, y_train_encoded)
```

Pour évaluer les performances du classifieur, nous effectuons une validation croisée 5 fois sur les données d'apprentissage. La validation croisée nous aide à obtenir une estimation plus précise des performances du modèle en l'évaluant sur différents sous-ensembles des données d'apprentissage :

```
xgb_cv_scores = cross_val_score(xgb_classifier, X_train_scaled, y_train_encoded,
cv=5)
print("XGBoost CV scores:")
print(xgb_cv_scores)
print("Mean CV score:", np.mean(xgb_scores))
```

Avec le modèle XGBoost entraîné, nous pouvons maintenant faire des prédictions sur l'ensemble de test. Nous décodons les étiquettes numériques prédites vers les étiquettes d'émotions originales en utilisant le 'LabelEncoder' :

```
y_pred_encoded = xgb_classifier.predict(X_test_scaled)
y_pred_xgb = label_encoder.inverse_transform(y_pred_encoded)
```

Nous évaluons les performances du classificateur XGBoost en générant un rapport de classification, qui comprend des mesures telles que la précision, le rappel et le score F1 :

```
print("Rapport :")
print(classification_report(y_test, y_pred_xgb))
```

Pour mieux visualiser les performances du classificateur, nous créons une matrice de confusion qui montre les vraies étiquettes par rapport aux étiquettes prédites :

	precision	recall	f1-score	support
ang	0.51	0.51	0.51	167
fru	0.32	0.31	0.32	204
hap	0.48	0.35	0.41	273
neu	0.37	0.37	0.37	231
sad	0.55	0.73	0.63	242
accuracy			0.45	1117
macro avg	0.45	0.45	0.45	1117
weighted avg	0.45	0.45	0.45	1117

FIGURE 3

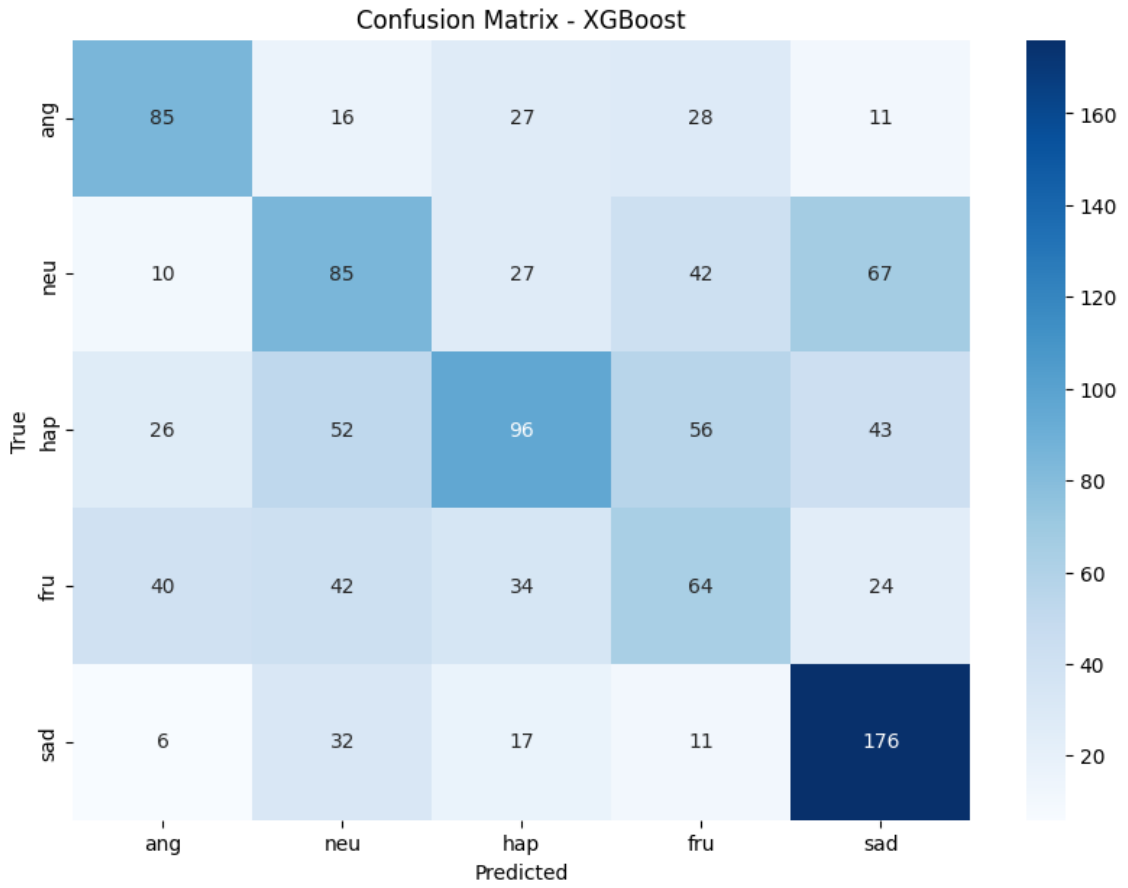


FIGURE 4 – Matrice de Confusion - XGBoost

```
cm = confusion_matrix(y_test, y_pred_xgb, labels=emotion_classes)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=emotion_classes,
yticklabels=emotion_classes)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - XGBoost')
plt.show()
```

La matrice de confusion nous aide à identifier les forces et les faiblesses du modèle en montrant dans quelle mesure il peut classer chaque classe d'émotions et où se produisent les erreurs de classification. Ces informations peuvent s'avérer précieuses pour interpréter les résultats et décider des améliorations ou des ajustements à apporter au modèle.

En conclusion, le classifieur XGBoost offre un moyen puissant et efficace d'effectuer la reconnaissance des émotions à partir des caractéristiques extraites. En combinant les forces du gradient boosting et des arbres de décision, il peut atteindre

des performances élevées dans des tâches de classification complexes. Ses performances peuvent être encore améliorées grâce à l'ajustement des hyperparamètres.

2.3.4 Réglage XGBoost - XGB optimisé

Dans cette partie, nous discuterons de l'optimisation du classifieur XGBoost en utilisant le réglage des hyperparamètres et la sélection des caractéristiques afin d'améliorer ses performances en matière de reconnaissance des émotions.

Tout d'abord, nous effectuons une recherche en grille pour trouver les meilleurs hyperparamètres pour le classificateur XGBoost. Nous spécifions une plage de valeurs pour les paramètres `learning_rate`, `max_depth` et `n_estimators` :

```
param_grid = {
    'learning_rate': [0.1, 0.2],
    'max_depth': [5, 7],
    'n_estimators': [100, 250],
}
```

Nous utilisons la fonction GridSearchCV de scikit-learn pour effectuer une validation croisée avec 5 'folds', en explorant toutes les combinaisons des valeurs des paramètres dans la grille :

```
grid_search = GridSearchCV(xgb_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train_encoded)
best_params = grid_search.best_params_
print("Best parameters found by grid search:")
print(best_params)
```

Nous obtenons le meilleur classificateur XGBoost avec les hyperparamètres optimaux :

```
best_xgb_classifier = grid_search.best_estimator_
```

```
{'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 250}
```

FIGURE 5 – Meilleure combinaison trouvée dans le domaine exploré

Ensuite, nous examinons les importances des caractéristiques obtenues par le meilleur classificateur XGBoost :

```
feature_importances = best_xgb_classifier.feature_importances_
indices = np.argsort(feature_importances)[::-1]

print("Feature importances:")
for i, index in enumerate(indices):
    print(f"{i + 1}. {X_train.columns[index]}: {feature_importances[index]}")
```

Pour optimiser les performances du classificateur, nous sélectionnons les k caractéristiques les plus importantes :

```
k = 15
selected_features = X_train.columns[indices[:k]]

X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]
```

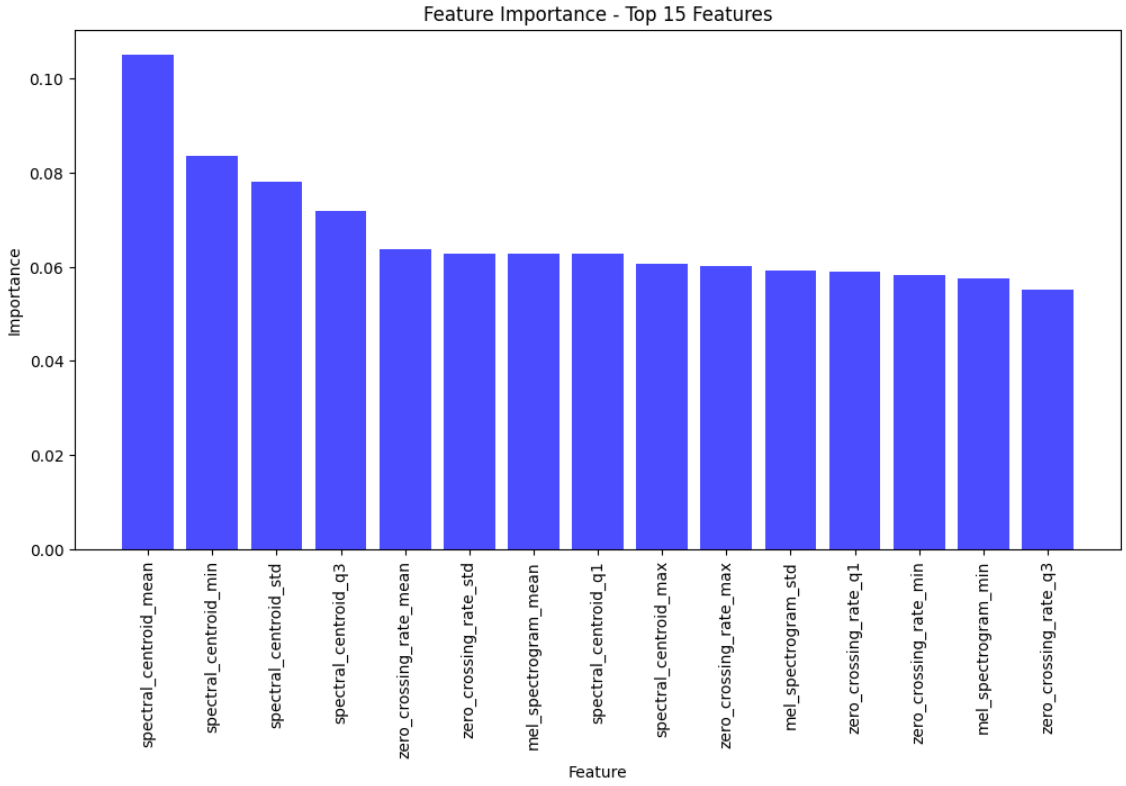


FIGURE 6

Nous réentraînons le meilleur classificateur XGBoost en utilisant uniquement les k premières caractéristiques et nous évaluons ses performances sur l'ensemble de test :

```
best_xgb_classifier.fit(X_train_selected, y_train_encoded)

y_pred_encoded_selected = best_xgb_classifier.predict(X_test_selected)
y_pred_xgb_selected = label_encoder.inverse_transform(y_pred_encoded_selected)

print("Classification Report (top k features):")
print(classification_report(y_test, y_pred_xgb_selected))
```

Enfin, nous visualisons les performances du classificateur XGBoost optimisé à l'aide d'une matrice de confusion :

```
cm = confusion_matrix(y_test, y_pred_xgb_selected, labels=emotion_classes)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=emotion_classes, yti
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - XGBoost (top k features)')
plt.show()
```

En conclusion, en optimisant les hyperparamètres et en sélectionnant les caractéristiques les plus importantes, nous pouvons améliorer les performances du classificateur XGBoost. Ce processus peut contribuer à réduire l'overfitting et à améliorer la généralisation du modèle à de nouvelles données.

Dans notre cas il y a une baisse légère par rapport au modèle précedent de la performance avec cette restriction aux 15 meilleurs variables, ce qui est peut-être expliqué par le fait que l'ensemble des paramètres explorés lors du réglage est assez réduit pour tenir compte des contraintes calculatoires.

2.4 Stacking ou Empilement

Dans cette section, nous discuterons de la mise en œuvre et des avantages de l'utilisation d'un ensemble d'empilage pour notre modèle de reconnaissance des émotions. L'empilement est une technique d'apprentissage d'ensemble qui combine plusieurs modèles de classification par l'intermédiaire d'un méta-classificateur. Les classificateurs de base sont formés sur la base de l'ensemble d'apprentissage complet, et le méta-classificateur est formé pour effectuer la prédiction finale sur la base des résultats des classificateurs de base.

Nous avons choisi l'empilement parce qu'il nous permet de combiner les forces des différents classificateurs, ce qui peut améliorer les performances globales de notre modèle de reconnaissance des émotions. En tirant parti des diverses forces des classificateurs de base, nous pouvons créer un modèle plus précis et plus robuste.

Tout d'abord, nous définissons les classificateurs de base qui seront combinés dans le processus d'empilement :

- Classifieur Random Forest (`random_forest_classifier`)
- Classifieur Support Vector Machine(SVM)(`svc_classifier`)
- Classifieur XGBoost optimisé avec les `k` meilleures features (`best_xgb_classifier`)

```
random_forest_classifier = RandomForestClassifier(n_estimators=100, random_state=1)
svc_classifier = SVC(kernel='rbf', C=1, gamma='scale',
probability=True, random_state=1)
```

Ensuite, nous créons un `StackingClassifier` à l'aide de scikit-learn. Les classificateurs de base sont transmis sous la forme d'une liste de tuples au paramètre `estimators`, et le classificateur final (`final_estimator`) est défini comme un classificateur XGBoost :

```
stacking_classifier = StackingClassifier(
    estimators=[
        ('random_forest', random_forest_classifier),
        ('svc', svc_classifier),
        ('xgboost', best_xgb_classifier)
    ],
    final_estimator=XGBClassifier(use_label_encoder=False, eval_metric='mlogloss',
)
```

Ensuite, nous créons un classificateur d'empilement à l'aide de scikit-learn. Les classificateurs de base sont passés sous la forme d'une liste de tuples au paramètre `estimators`, et le classificateur final (`final_estimator`) est défini comme un classificateur XGBoost :

```
stacking_classifier.fit(X_train_selected, y_train_encoded)
y_pred_encoded = stacking_classifier.predict(X_test_selected)
y_pred_stacking = label_encoder.inverse_transform(y_pred_encoded)
```

Après avoir obtenu les prédictions, nous évaluons les performances du classificateur d'empilement :

```
print("Classification Report (Stacking):")
print(classification_report(y_test, y_pred_stacking))
```

Classification Report (Stacking):

	precision	recall	f1-score	support
ang	0.50	0.46	0.47	167
fru	0.25	0.39	0.31	204
hap	0.39	0.23	0.29	273
neu	0.29	0.32	0.31	231
sad	0.62	0.58	0.60	242
accuracy			0.39	1117
macro avg	0.41	0.40	0.40	1117
weighted avg	0.41	0.39	0.39	1117

FIGURE 7

En conclusion, l'empilement est une technique puissante qui permet de combiner les forces de plusieurs classificateurs et d'améliorer potentiellement les performances globales de notre modèle de reconnaissance des émotions. L'approche de l'empilement nous permet d'exploiter les diverses forces des classificateurs de base pour créer un modèle plus précis et plus robuste. Elle permet d'incorporer différents types de classificateurs et leurs caractéristiques uniques, ce qui peut conduire à une meilleure généralisation et à des performances de prédiction améliorées.

Dans ce cas là aussi, on a une baisse de la performance en comparaison avec chacun des modèles pris seul, pour les mêmes raisons de contraintes calculatoires pour le reglage des parametres pour ces modeles qui est tres couteux (la taille de la base de données est-peut être aussi une des raisons, puisque ce genre de techniques nécessitent en général un nombre plus important d'entrées pour l'entraînement compare aux modèles pris seuls)

3 Résultats

3.1 Remarque : Résultats XGB et Stacking

Les résultats de ces modèles ont été discutés dans la section précédente (Méthodologie) pour pouvoir s'en servir au fur et à mesure du réglage.

3.2 CNN

3.2.1 Rapport de classification

Classification Report:

	precision	recall	f1-score	support
ang	0.41	0.78	0.53	200
fru	0.29	0.42	0.34	200
hap	0.33	0.01	0.01	200
neu	0.36	0.23	0.28	200
sad	0.67	0.64	0.66	200
accuracy			0.42	1000
macro avg	0.41	0.42	0.37	1000
weighted avg	0.41	0.42	0.37	1000

FIGURE 8 – Rapport de classification du modèle CNN

3.2.2 Matrice de confusion

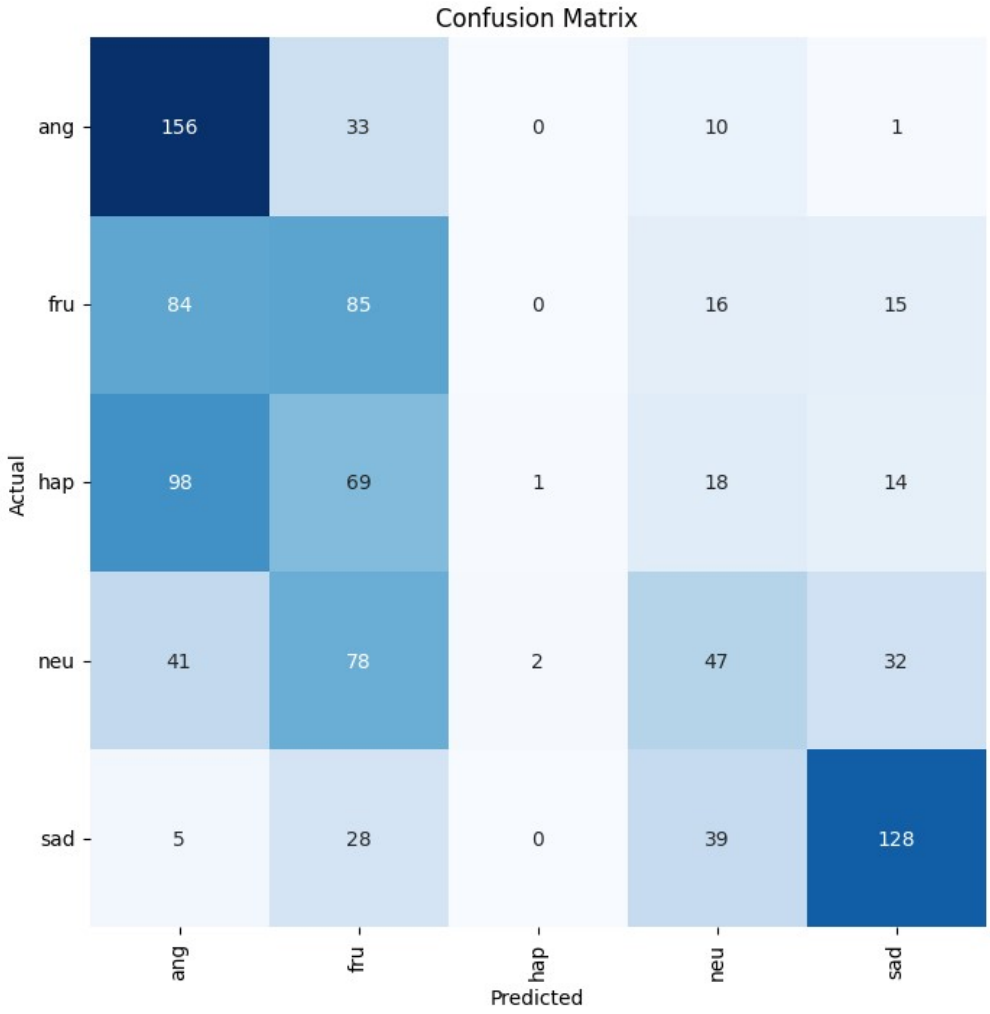


FIGURE 9 – Matrice de confusion du modèle CNN

3.2.3 Interprétation des résultats

Une performance globale relativement bonne (comparée à celles des modèles testés précédemment). Nous remarquons cependant que les échantillons de la classe 'hap' ne sont presque jamais prédits, sans qu'aucune raison valable n'apparaisse à première vue pour cette classe (créée par la fusion de 'hap' et 'exc' précédemment). Ce phénomène n'est rencontré que pour ce modèle.

3.3 Random Forest

3.3.1 Rapport de classification

Classification Report:

	precision	recall	f1-score	support
ang	0.49	0.54	0.52	167
fru	0.30	0.35	0.32	204
hap	0.49	0.26	0.34	273
neu	0.36	0.37	0.36	231
sad	0.58	0.75	0.66	242
accuracy			0.45	1117
macro avg	0.44	0.46	0.44	1117
weighted avg	0.45	0.45	0.44	1117

FIGURE 10 – Rapport de classification du modèle Random Forest

3.3.2 Analyse du rapport de classification

Il s'agit d'un rapport de classification pour un modèle qui prédit les émotions. Le rapport montre la précision, le rappel et le score F1 pour chaque état émotionnel, ainsi que la précision globale, et les moyennes macro et pondérées.

En regardant le rapport, la précision pour prédire 'ang' et 'sad' est relativement élevée, ce qui indique que lorsque le modèle prédit ces états émotionnels, il a généralement raison. Cependant, le rappel pour prédire 'hap' est assez faible, indiquant que le modèle n'est pas très bon pour identifier cet état émotionnel.

Le score F1 est une moyenne pondérée de la précision et du rappel, où la meilleure valeur est 1 et la pire est 0,5. Le score F1 pour 'sad' est le plus élevé, ce qui indique que le modèle fonctionne le mieux pour cet état émotionnel. En revanche, le score F1 pour 'fru' est le plus bas, indiquant que le modèle fonctionne le moins bien pour cet état émotionnel.

La précision du modèle est de 0,45, ce qui signifie qu'il prédit correctement l'état émotionnel du texte environ 45du temps.

Les moyennes macro et pondérées tiennent compte du nombre d'échantillons dans chaque classe. La moyenne macro est calculée en prenant la moyenne de la précision, du rappel et du score F1 pour chaque classe. La moyenne pondérée est

calculée en prenant la moyenne pondérée de la précision, du rappel et du score F1 pour chaque classe, où les poids sont le nombre d'échantillons dans chaque classe. Dans ce rapport, les moyennes macro et pondérées sont assez similaires, ce qui indique qu'il n'y a pas de déséquilibre significatif entre les classes.

3.3.3 Matrice de confusion

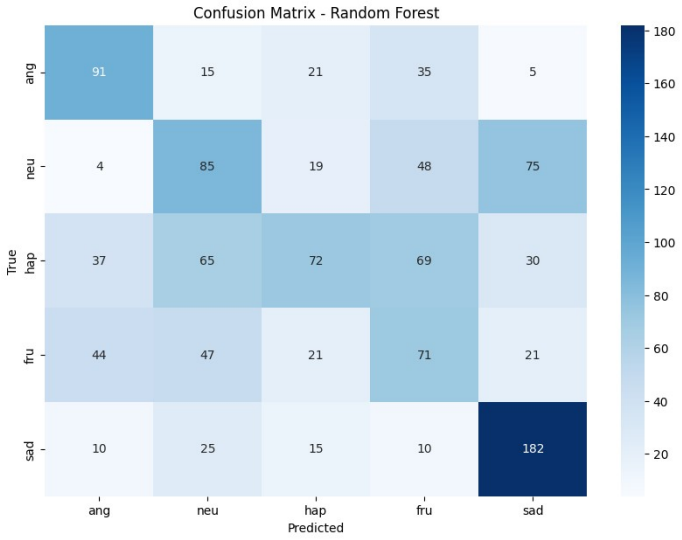


FIGURE 11 – Matrice de confusion du modèle Random Forest

3.3.4 Interprétation de la matrice de confusion

On remarque que les coefficients diagonaux de la matrice de confusion sont plus grands que les autres coefficients, cela signifie que le modèle est relativement performant et qu'il prédit correctement la plupart des classes. En d'autres termes, il a une haute précision et une haute sensibilité pour la plupart des classes. Toutefois, cela ne signifie pas nécessairement que le modèle est parfait, car il peut encore y avoir des erreurs de prédiction pour certaines classes.

En plus, on remarque que il y une confusion importante entre "happy" et "neu", happy" et "fru", "neu" et sad". Cela revient à la performence du modèle qu'on peut améliorer, soit en choissisant judicieusement les features et en jouant sur les hyperparamètres.

4 Conclusion

En conclusion, cette étude visait à explorer divers modèles et techniques d'apprentissage automatique pour prédire les émotions à partir de signaux vocaux. Nous avons étudié l'importance de différentes caractéristiques, telles que les MFCC, les mel-spectrogrammes, les centroïdes spectraux et les taux de passage à zéro, et nous avons utilisé plusieurs modèles de classification, notamment Random Forest, CNN, XGBoost et Stacking. Nous avons également procédé à l'ajustement des hyperparamètres et à la sélection des caractéristiques afin d'optimiser davantage nos modèles.

Nos résultats indiquent que les différents modèles attribuent des niveaux d'importance variables aux caractéristiques, probablement en raison de leurs mécanismes d'apprentissage uniques et des modèles qu'ils capturent dans les données. En combinant les connaissances de plusieurs modèles, nous avons pu acquérir une compréhension plus complète de l'ensemble de données et des relations sous-jacentes entre les caractéristiques et les classes d'émotions cibles.

Une possibilité serait de se concentrer sur l'intégration de caractéristiques supplémentaires, telles que les caractéristiques prosodiques ou linguistiques, et sur l'expérimentation d'autres techniques, telles que l'apprentissage par transfert, afin d'améliorer la précision de la prédiction et la généralisabilité. En fin de compte, les progrès de la reconnaissance des émotions à partir de la parole ont le potentiel de révolutionner diverses applications, y compris le suivi de la santé mentale, le service à la clientèle et l'interaction homme-machine, entre autres.

5 Réferences bibliographiques

https ://sail.usc.edu/iemocap/index.html
https ://medium.com/heuristics/audio-signal-feature-extraction-and-clustering-935319d2225
https ://www.kaggle.com/code/shivamburnwal/speech-emotion-recognition
https ://www.sciencedirect.com/science/article/pii/S1877050920318512
https ://www.projectpro.io/article/speech-emotion-recognition-project-using-machine-learning/573