Université Mohammed V de Rabat Ecole Nationale Supérieure d'Informatique et d'Analyse et des systèmes

Rapport de projet d'Intelligence Artificielle

Option

e-Management & Business Intelligence

Sujet

Reconnaissance des genres de musique

Soutenu par :

ASSABAA Ghassan

EL YOUSFI Ismail

Encadré par :

Mme BENBRAHIM Houda

Année universitaire 2019 - 2020

Introduction :

La recherche d'informations musicales (MIR) est un domaine interdisciplinaire traitant de l'analyse du contenu musical en combinant des aspects du traitement du signal, de l'apprentissage automatique et de la théorie musicale. MIR permet aux algorithmes informatiques de comprendre et de traiter les données musicales de manière intelligente. Reconnaissance des genres musicaux (MGR) est l'un des sous-domaines les plus importants du MIR. Le genre de musique est défini comme un style de musique expressif incorporant des tons instrumentaux ou vocaux de manière structurée appartenant à un ensemble de conventions. La reconnaissance automatique des genres musicaux est un problème intéressant dans le contexte du MIR car il permet à des systèmes d'effectuer des recommandations musicales basées sur le contenu, organiser des bases de données musicales et découvrir des collections de médias.

La classification des genres musicaux est un problème populaire dans l'apprentissage automatique et peut être utilisé avec plusieurs applications pour marquer chaque chanson dans un énorme corpus musical avec le genre et le sous-genre qui peuvent ensuite être utilisé pour identifier des chansons similaires. Traditionnellement, nous avons extrait des fonctionnalités de la musique audio comme MFCC et les avons utilisées comme point pour une tâche de classification.

Ce projet consiste donc de créer un programme en Python qui va faire la reconnaissance du genre musical d'un extrait audio Un réseau de neurones sera donc entrainé, pour faire cette reconnaissance, en utilisant un échantillon de fichiers audio. Un modèle d'extraction d'attributs simple sera utilisé.

Le projet fut réparti en 4 grandes étapes :

- Extraction des variables
- Analyse de données
- Conception du modèle
- Interface graphique

Table des matières

1. Chapitre 1 : Extraction des variables

L'extraction des variables est un élément très important dans l'analyse et la recherche de relations entre différentes choses. Les données audio fournies ne peuvent pas être comprises directement par les modèles pour les convertir en un format compréhensible. C'est un processus qui explique la plupart des données mais de manière compréhensible.

L'extraction des variables est requise pour les algorithmes de classification, de prédiction et de recommandation. Pour ce faire, nous présenterons alors les variables à extraire des fichiers musicaux qui nous aideront à classer les fichiers musicaux dans différents genres.

Le signal audio est un signal tridimensionnel dans lequel trois axes représentent le temps, l'amplitude et la fréquence.

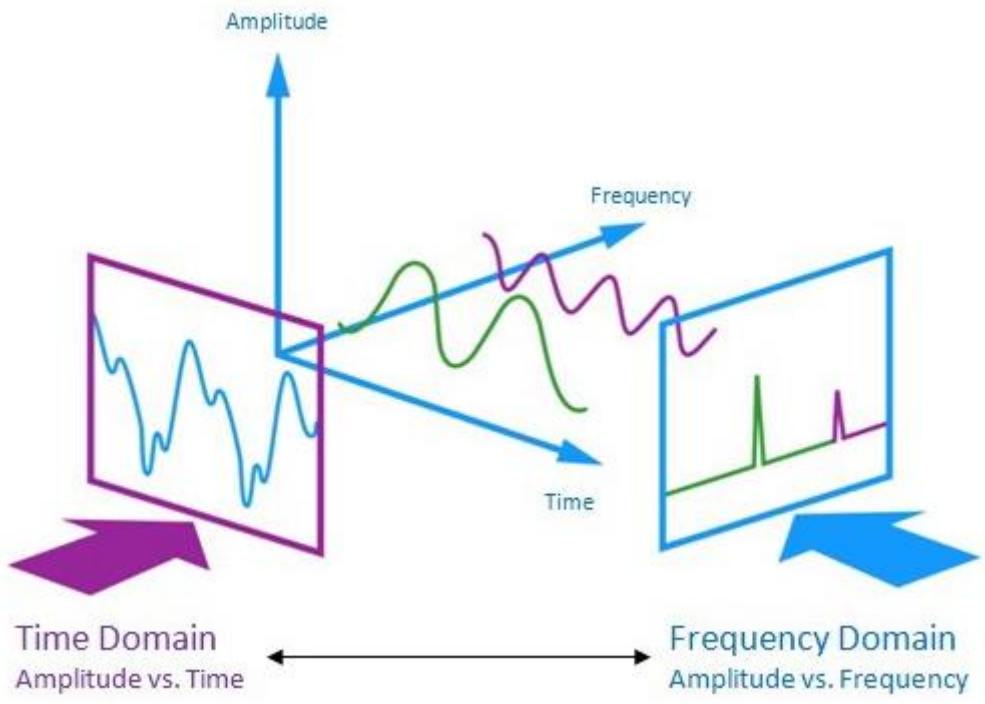


Figure 1 : Signal audio

1. MFCC

Cette variable est l'une des méthodes les plus importantes pour extraire une fonction d'un signal audio et est utilisée principalement lorsque vous travaillez sur des signaux audio. Les coefficients spectraux de fréquence de mélange (MFCC) d'un signal sont un petit ensemble de caractéristiques (généralement environ 10-20) qui décrivent de manière concise la forme globale d'une enveloppe spectrale.

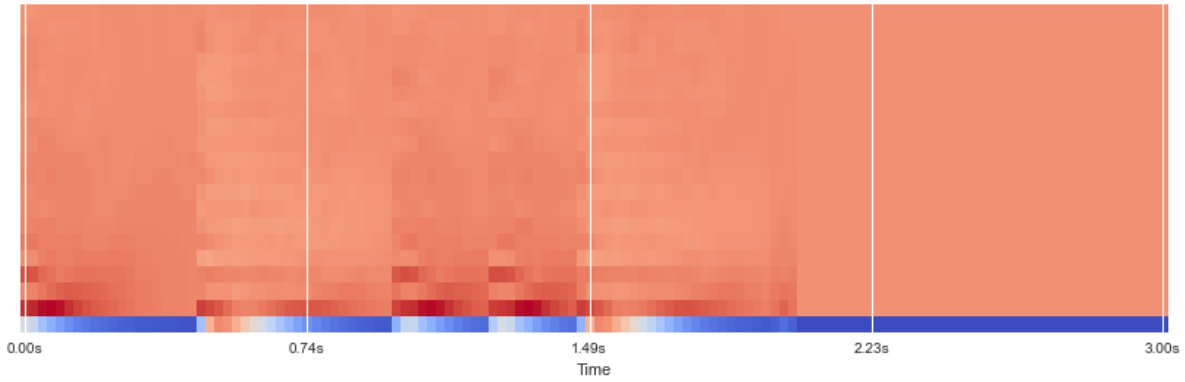


Figure 2 : MFCC

2. ZCR (Zero Crossing Rate)

Le taux de passage par zéro est le taux de changements de signe le long d'un signal, c'est-à-dire le taux auquel le signal passe du positif au négatif ou inversement. Cette fonction a été largement utilisée à la fois dans la reconnaissance vocale et la récupération d'informations musicales. Il a généralement des valeurs plus élevées pour les sons hautement percutants comme ceux du métal et du rock.

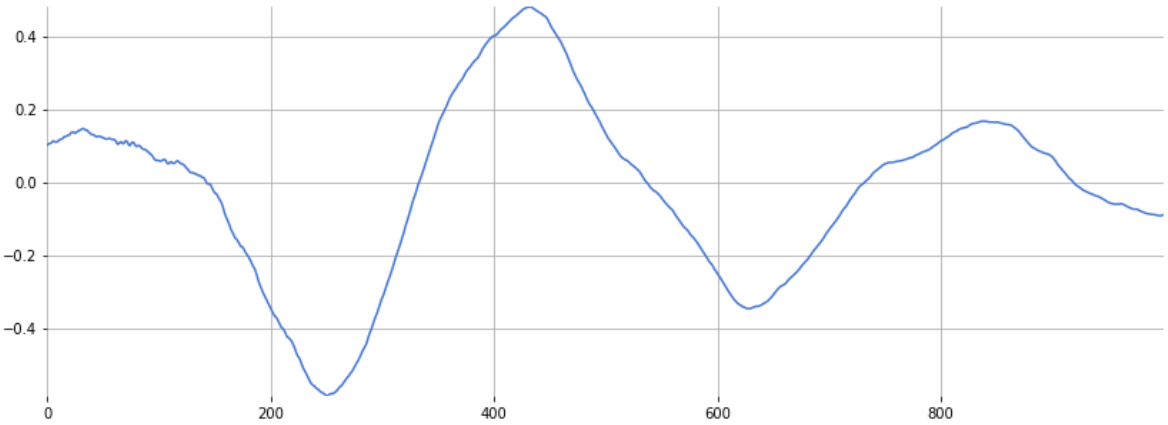


Figure 3 : ZCR

3. Chroma

Les caractéristiques de chrominance sont une représentation intéressante et puissante pour la musique audio dans laquelle le spectre entier est projeté sur 12 cases représentant les 12 demi-tons distincts (ou chroma) de l'octave musicale. Étant donné que, en musique, les notes situées exactement à une octave sont perçues comme particulièrement similaires, connaître la distribution de la chrominance même sans la fréquence absolue (c'est-à-dire l'octave d'origine) peut donner des informations musicales utiles sur l'audio - et peut même révéler une similitude musicale perçue qui n'est pas apparent dans les spectres originaux.

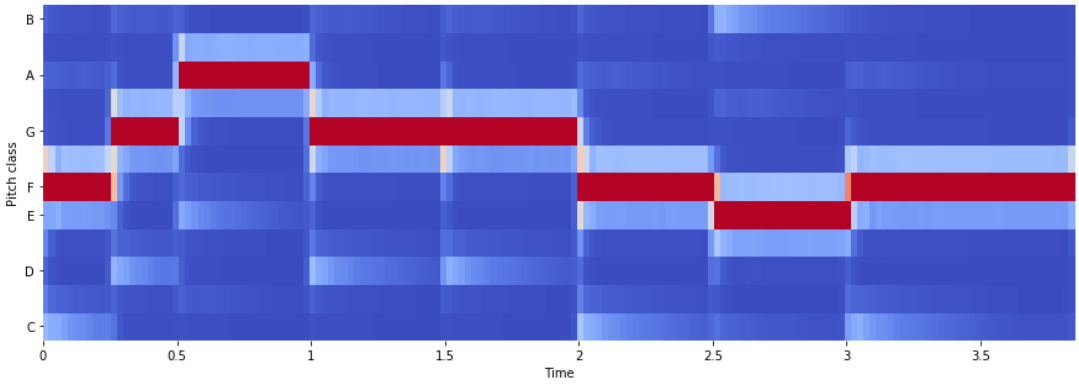


Figure 4 : Chroma

4. RMSE (Root Mean Square Error)

L'erreur quadratique moyenne (RMSE) est l'écart type des résidus (erreurs de prédiction). Les résidus sont une mesure de l'éloignement des points de données de la ligne de régression; RMSE est une mesure de la répartition de ces résidus. En d'autres termes, il vous indique la concentration des données autour de la ligne de meilleur ajustement. L'erreur quadratique moyenne est couramment utilisée en climatologie, prévision et analyse de régression pour vérifier les résultats expérimentaux.

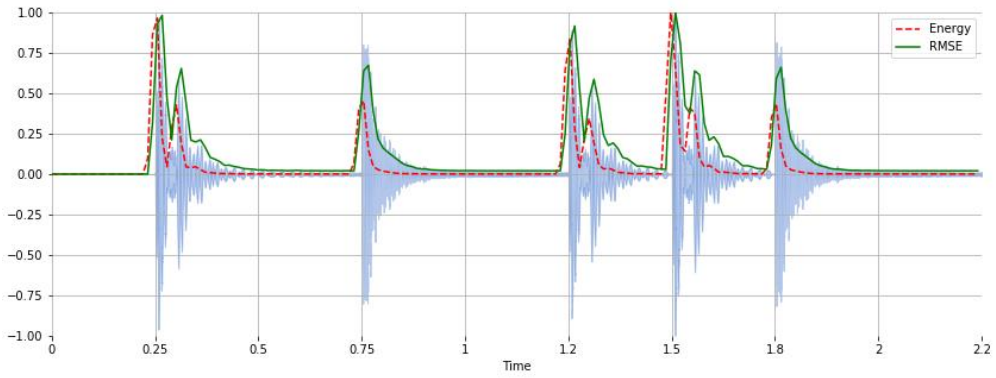


Figure 5 : RMSE

5. Spectral

a. *Spectral centroid*

Il indique où se situe le «centre de masse» d'un son et est calculé comme la moyenne pondérée des fréquences présentes dans le son. Si les fréquences en musique sont les mêmes partout, le centroïde spectral serait autour d'un centre et s'il y a des fréquences élevées à la fin du son, le centroïde serait vers sa fin.

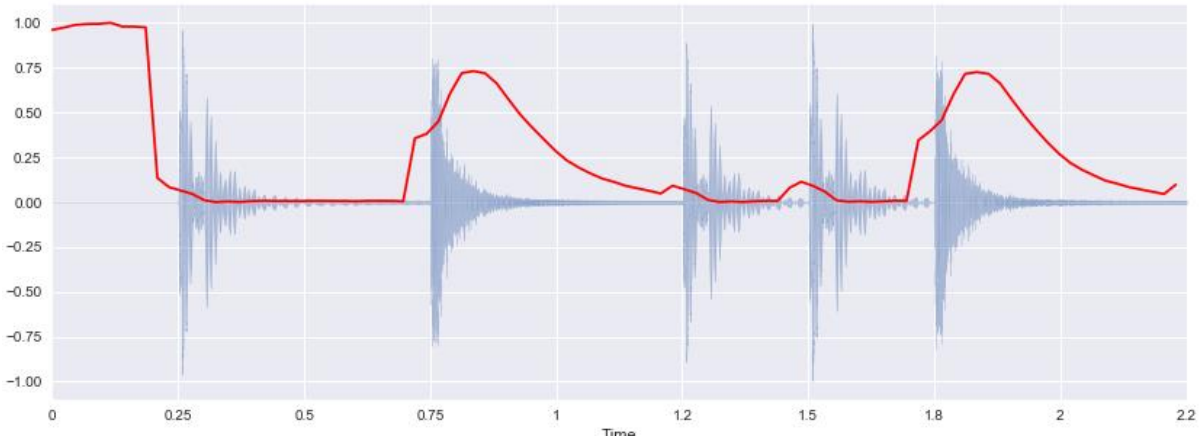


Figure 6 : spectral centroid

b. *Spectral contrast*

Le contraste spectral prend en compte le pic spectral, la vallée spectrale et leur différence dans chaque sous-bande de fréquence.

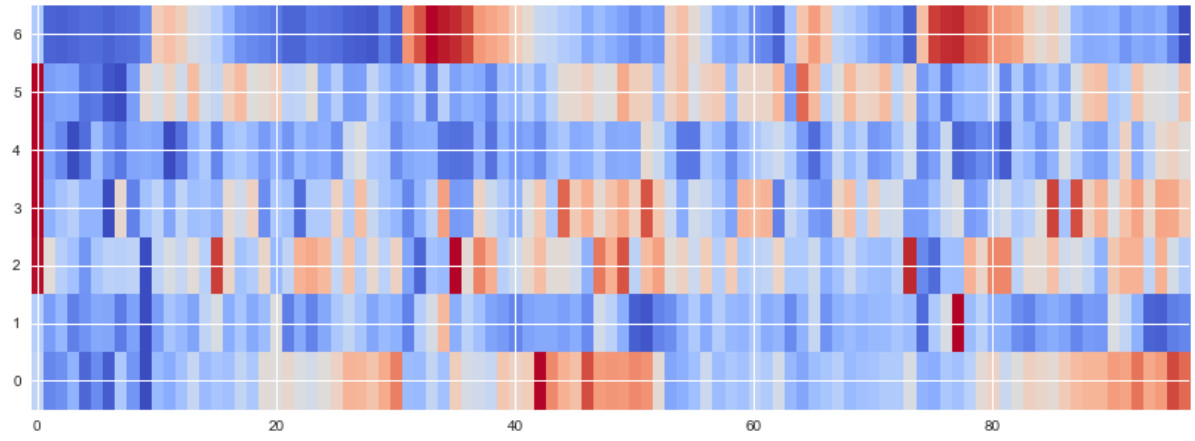


Figure 7 : spectral contrast

c. *Spectral bandwith*

La bande passante est le principal critère d'évaluation de la qualité des appareils audio tels que les enceintes, casques et autres systèmes pourvus de haut-parleurs. Tout son est une vibration de l'air, et l'une de ses caractéristiques principales est sa fréquence, c'est-à-dire son nombre de vibrations par seconde, mesuré en hertz (Hz). Plus ce nombre est élevé, plus le son est aigu. Une oreille humaine moyenne entend les sons de fréquences comprises entre 20 Hz (son très grave) et 20 000 Hz (son très aigu).

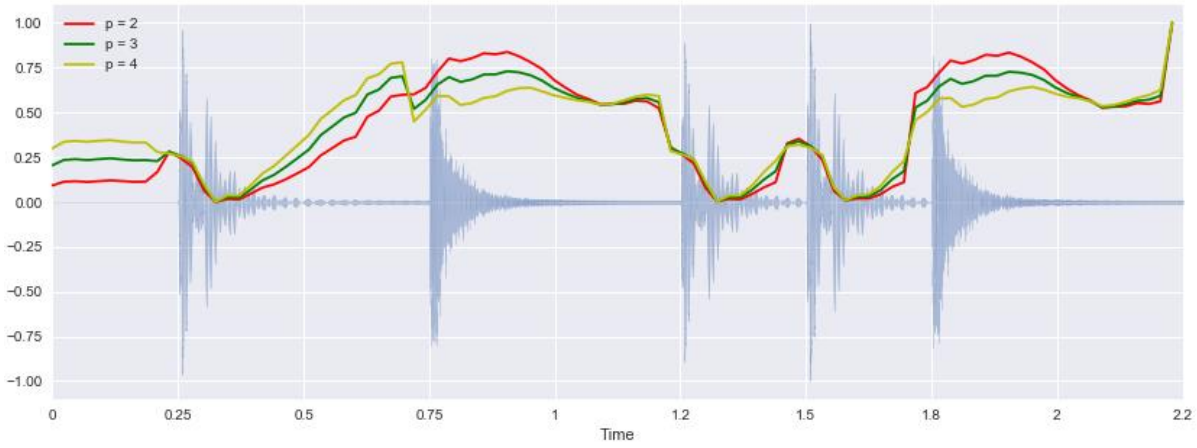


Figure 8 : spectral bandwidth

d. *Spectral Rollof*

L'atténuation spectrale est la fréquence en dessous de laquelle un pourcentage spécifié de l'énergie spectrale totale, par ex. 85%, pond.

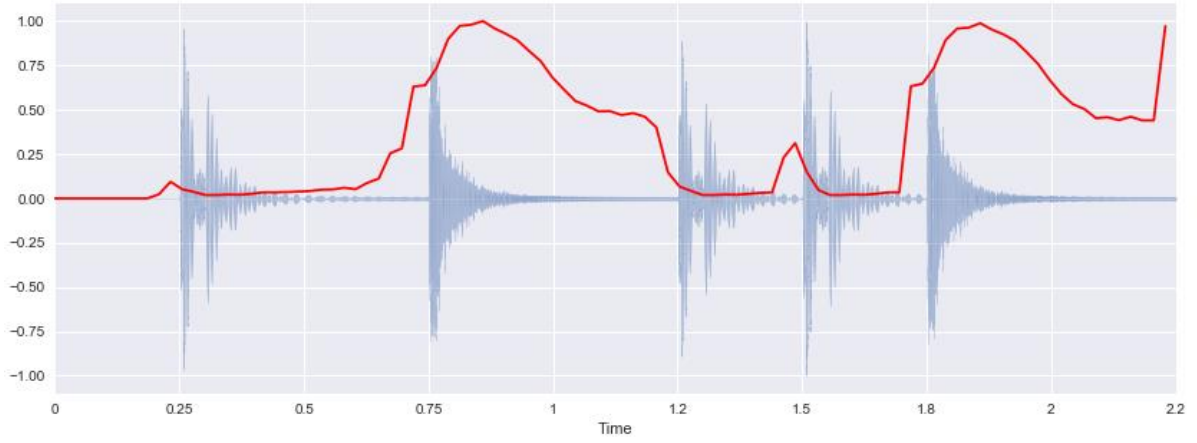


Figure 9 : spectral Rollof

2. Chapitre 2 : Analyse des données

Il est nécessaire de décrire et explorer les données avant d'en tirer de quelconques lois ou modèles prédictifs. Il est donc nécessaire d'extraire les informations pertinentes avant d'entamer la partie de la conception de notre modèle, de reconnaissance de genre de music.

a. Statistiques descriptives :

La boîte à moustaches résume seulement quelques indicateurs de position du caractère étudié (médiane, quartiles, minimum, maximum ou déciles). Ce diagramme est utilisé principalement pour comparer un même caractère dans deux populations de tailles différentes.

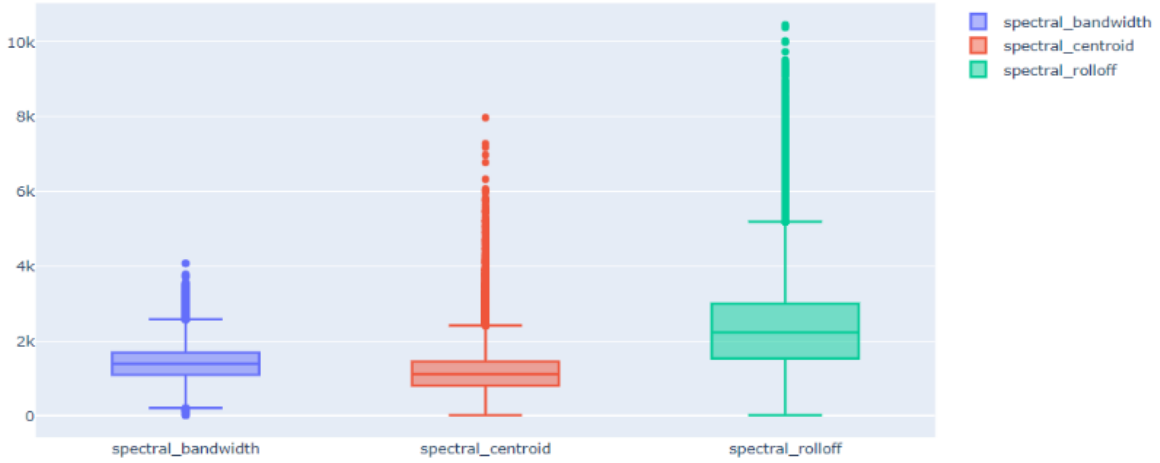


Figure 10 : Boîte à moustache de quelques variables extraites d'un audio

b. Analyse en composantes principales :

Une matrice de corrélation est utilisée pour évaluer la dépendance entre plusieurs variables en même temps. Le résultat est une table contenant les coefficients de corrélation entre chaque variable et les autres.

Pour notre cas, on a montré la dépendance entre nos variables extraites depuis les fichiers audio, pour tous les outils de calcul qui seront utilisé après (moyenne, médiane, kurtosis …).

			chroma_cens kurtosis 01	02	03	04	05	06	07	08	09	10	...	tonnetz std 04	05
chroma_cens	kurtosis	01	1.000000	0.168040	0.072936	0.114748	0.087447	0.055913	0.093210	0.046958	0.062980	0.135635	...	-0.021042	-0.016246
		02	0.168040	1.000000	0.256776	0.581978	0.563089	0.374696	0.670683	0.249413	0.277566	0.386960	...	-0.047317	-0.038900
		03	0.072936	0.256776	1.000000	0.696866	0.243232	0.159603	0.208861	0.093052	0.126101	0.108099	...	-0.046026	-0.030043
		04	0.114748	0.581978	0.696866	1.000000	0.590232	0.412467	0.549015	0.193489	0.183819	0.093965	...	-0.028608	-0.021130
		05	0.087447	0.563089	0.243232	0.590232	1.000000	0.682713	0.530515	0.191569	0.192796	0.222326	...	-0.036079	-0.027037
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
zcr	mean	01	-0.001692	-0.001086	-0.005155	-0.000912	-0.004149	-0.000639	-0.000600	-0.001312	0.006950	-0.006680	...	-0.267163	-0.295029
	median	01	0.001103	0.003289	0.000021	0.002799	0.000900	0.005873	0.004010	0.004069	0.015165	-0.000849	...	-0.263548	-0.297247
	min	01	0.006344	0.004314	0.000966	0.000096	-0.000729	-0.000198	0.000075	-0.000354	0.005019	0.006823	...	-0.061446	-0.098449
	skew	01	0.025801	0.016010	0.019725	0.003503	0.048093	0.012586	0.017420	0.023715	0.030556	0.022298	...	0.181227	0.215362
	std	01	-0.009808	-0.011634	-0.016049	-0.008062	-0.009609	-0.009060	-0.007351	-0.009298	-0.010939	-0.016175	...	-0.105214	-0.104477

Figure 11 : Matrice de corrélation

Ensuite viens l'étape de trouver le nombre de dimensions optimales qui nous garantira la collection le maximum d'informations. Pour cela on utilisera le graphique des valeurs propres ainsi que le tableau de la variance totale expliquée.

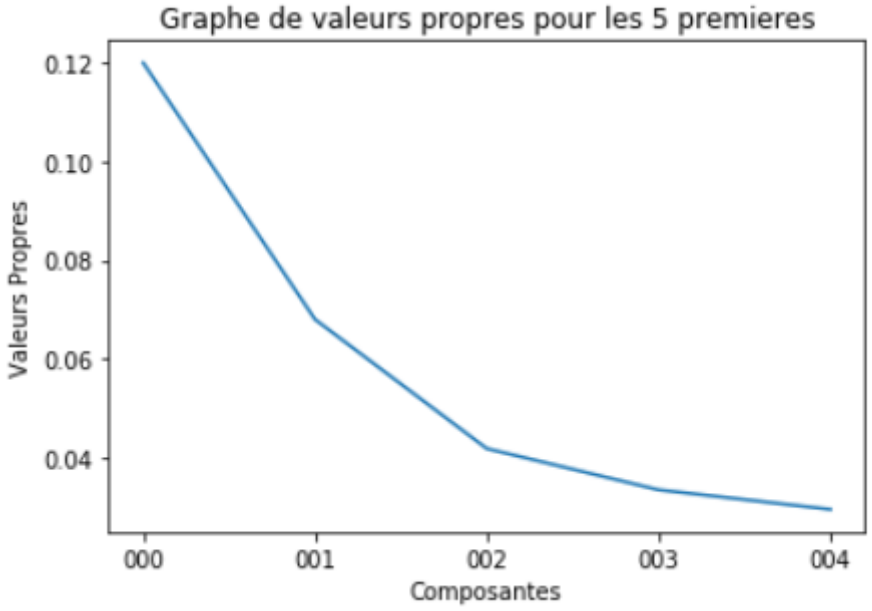


Figure 12 : Graphique des valeurs propres

Sous contraintes de visualisation, on a pu visualiser que le graphe des cinq premières valeurs. Donc on ne peut pas prendre une décision concernant le nombre de dimension qu'on peut utiliser pour notre modèle.

	Composante	Total	% de la variance	% cumules
0	000	1.198121e-01	11.981209	11.981209
1	001	6.802443e-02	6.802443	18.783652
2	002	4.195006e-02	4.195006	22.978658
3	003	3.362508e-02	3.362508	26.341166
4	004	2.968533e-02	2.968533	29.309699
...	...	...	...	...
513	513	1.961277e-06	0.000196	99.999533
514	514	1.752000e-06	0.000175	99.999709
515	515	1.334375e-06	0.000133	99.999842
516	516	8.445070e-07	0.000084	99.999926
517	517	7.359865e-07	0.000074	100.000000

Figure 13 : Table des variances totales expliquées

Et afin de trouver le nombre de dimensions optimal afin de garantir un bon pourcentage d'informations expliqués, On a créé un petit script permettant ainsi de visualiser tout ça.

Entrée [12]:
```
information = 0
i=0
while information <= 80:
    information += eigenVal[i]*100
    i+=1
print("\n\nNombre de facteurs necessaires pour nous garantir 80% de l'information expliquee : ",i," \n")
while information <= 90:
    information += eigenVal[i]*100
    i+=1
print("\n\nNombre de facteurs necessaires pour nous garantir 90% de l'information expliquee : ",i," \n")
while information <= 95:
    information += eigenVal[i]*100
    i+=1
print("\n\nNombre de facteurs necessaires pour nous garantir 95% de l'information expliquee : ",i," \n")
```

```
Nombre de facteurs necessaires pour nous garantir 80% de l'information expliquee :  85


Nombre de facteurs necessaires pour nous garantir 90% de l'information expliquee :  151


Nombre de facteurs necessaires pour nous garantir 95% de l'information expliquee :  215
```

Figure 14 : Script visualisant le nombre de facteur

L'analyse PCA appliquée sur nos données est donc ensuite validée en appliquant la fonction de « fiting ». Nous donnant ainsi un résultat encore plus précis.

```
pComponents85 = PCA85.fit_transform(X_train)
pComponents151 = PCA151.fit_transform(X_train)
pComponents215 = PCA215.fit_transform(X_train)
```

```
print('Pour 85 facteurs : ',PCA85.explained_variance_ratio_.sum(),' de l information est expliquee\n')
print('Pour 151 facteurs : ',PCA151.explained_variance_ratio_.sum(),' de l information est expliquee\n')
print('Pour 215 facteurs : ',PCA215.explained_variance_ratio_.sum(),' de l information est expliquee\n')
```

```
Pour 85 facteurs :  0.8090947500557024  de l information est expliquee

Pour 151 facteurs :  0.9078570837006089  de l information est expliquee

Pour 215 facteurs :  0.955472083818175  de l information est expliquee
```

Figure 15 : PCA

3. Chapitre 3 : Conception du modèle

a. Deep Neural Networks (DNNs)

Les réseaux de neurones sont un ensemble d'algorithmes, modélisés d'après le cerveau humain, qui sont conçus pour reconnaître les modèles. Ils interprètent les données sensorielles à travers une sorte de perception de machine, d'étiquetage ou de regroupement des entrées brutes. Les variables qu'ils reconnaissent sont numériques, contenus dans des vecteurs, dans lesquels toutes les données du monde réel, que ce soit des images, du son, du texte ou des séries temporelles, doivent être traduites.

Les réseaux de neurones profonds (DNN) sont généralement des réseaux de transmission directe (FFNN) dans lesquels les données circulent de la couche d'entrée vers la couche de sortie sans reculer et les liens entre les couches sont à sens unique dans le sens aller et ne touchent jamais un nœud encore.

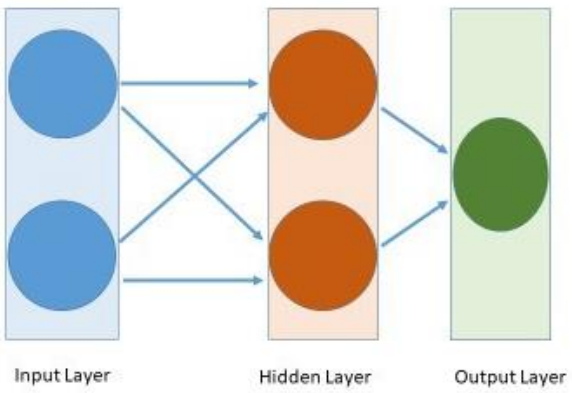


Figure 16 : Deep neural networks (DNNs)

Ils sont utilisés pour regrouper et classer. Vous pouvez les considérer comme une couche de clustering et de classification au-dessus des données que vous stockez et gérez. Ils aident à regrouper les données non étiquetées en fonction des similitudes entre les exemples d'entrées, et ils classent les données lorsqu'ils ont un ensemble de données étiqueté pour s'entraîner.

b. Architecture de notre modèle

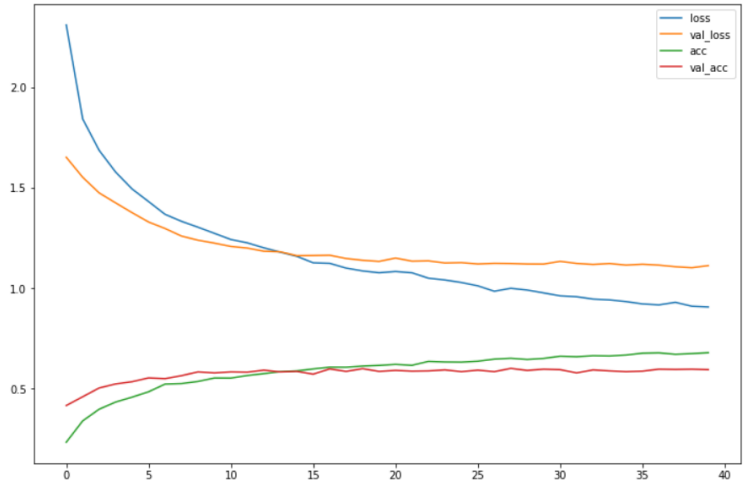
Pour l'architecture de notre modèle, on a effectué deux architectures différentes afin de choisir au final celle avec le résultat le plus important et la plus grande précision.

La première architecture est un « Decoder », ce dernier utilise l'architecture de notre second modèle qui est un « Encoder ». Il est caractérisé par une augmentation de nombre de neurones dans l'une ou plusieurs des couches cachées.

```
model = Sequential()
model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))

model.add(Dropout(0.6))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(8, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

Figure 17 : Architecture du Decoder model

La première architecture nous a renvoyé un modèle avec une précision de 0.59375




```
800/800 [==============================] - 0s 52us/sample - loss: 1.2471 - acc: 0.5938
Test loss: 1.2470972847938537
Test accuracy: 0.59375
```

Figure 18 : Graphe du score du premier modèle

On peut aussi voir sur la « Heat map » ci-dessous la première répartition des genres musicaux qu'à donner notre premier modèle.

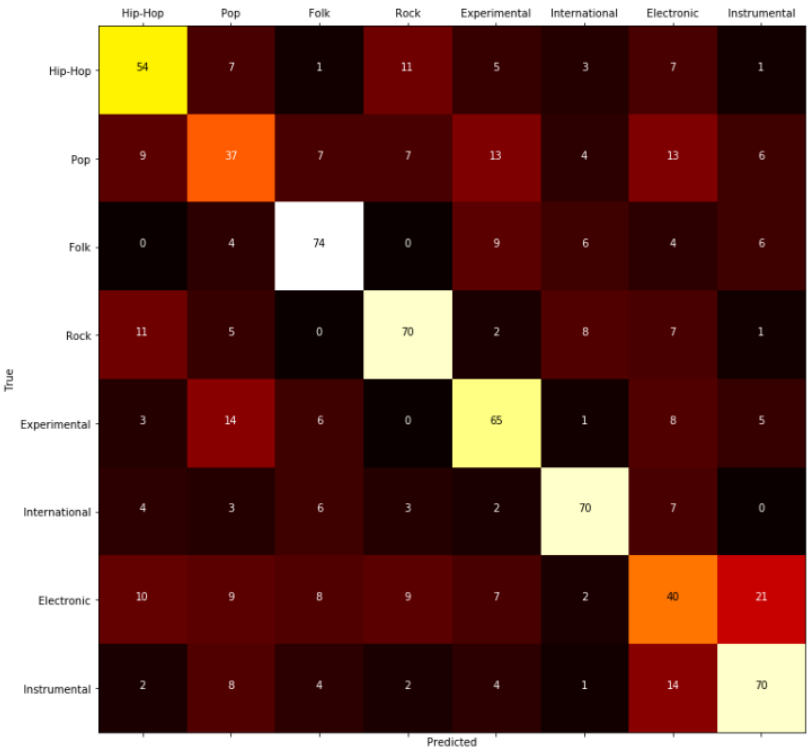


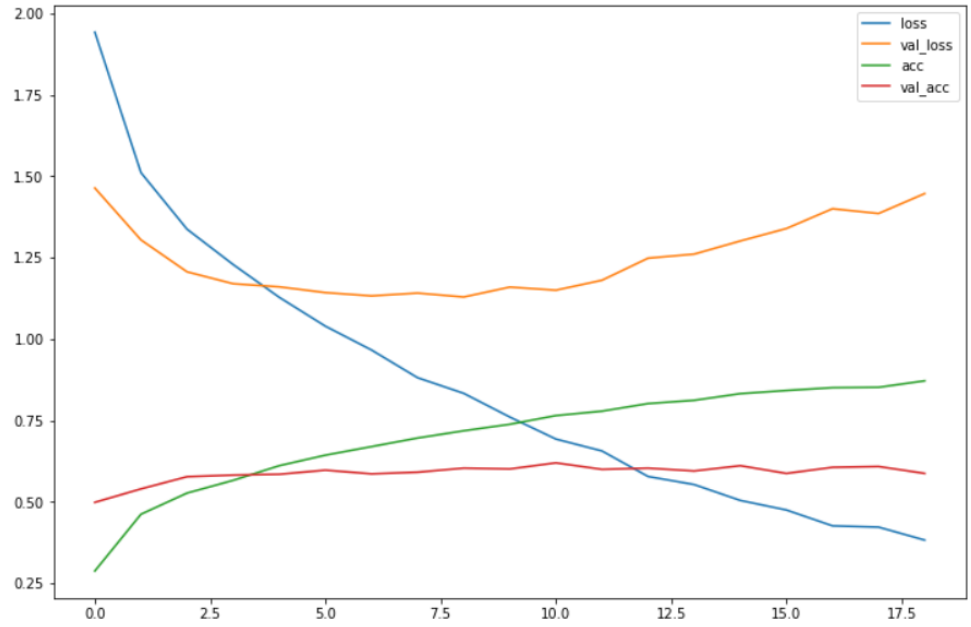
Figure 19 : Heat map du premier modèle

Pour l'architecture du deuxième modèle, ce dernier est de type « Encoder ». Il est caractérisé par la réduction des nombres des neurones chaque fois qu'on se déplace d'une couche à une autre à partir de la couche d'entrée en direction de la couche de sortie.

```
model2 = Sequential()
model2.add(Dense(512, input_dim=X_train.shape[1], activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(256, activation='relu'))
model2.add(Dropout(0.4))
model2.add(Dense(256, activation='relu'))
model2.add(Dropout(0.6))
model2.add(Dense(8, activation='softmax'))

model2.compile(optimizer='nadam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model2.summary()
```

Figure 20 : Architecture de l'Encoder Model

Qui nous générera lui aussi son graphe de score avec une meilleure précision qui est de 0,6.




```
800/800 [==============================] - 0s 81us/sample - loss: 1.6690 - acc: 0.6000
Test loss: 1.6690121126174926
Test accuracy: 0.6
```

Figure 21 : Graphe du score du premier modèle

Et on peut aussi voir que la répartition des genres musicaux que ce modèle applique est bien meilleure que le modèle précédent.

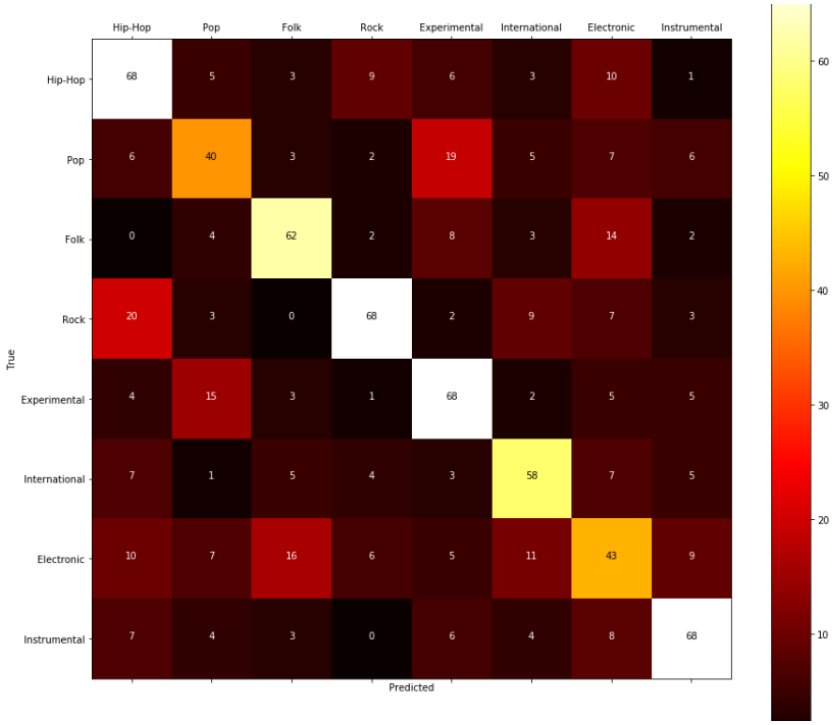


Figure 22 : Heat map du deuxième modèle

4. Chapitre 4 : Interface graphique

L'interface graphique a été implémentée en utilisant la bibliothèque PyQt5.

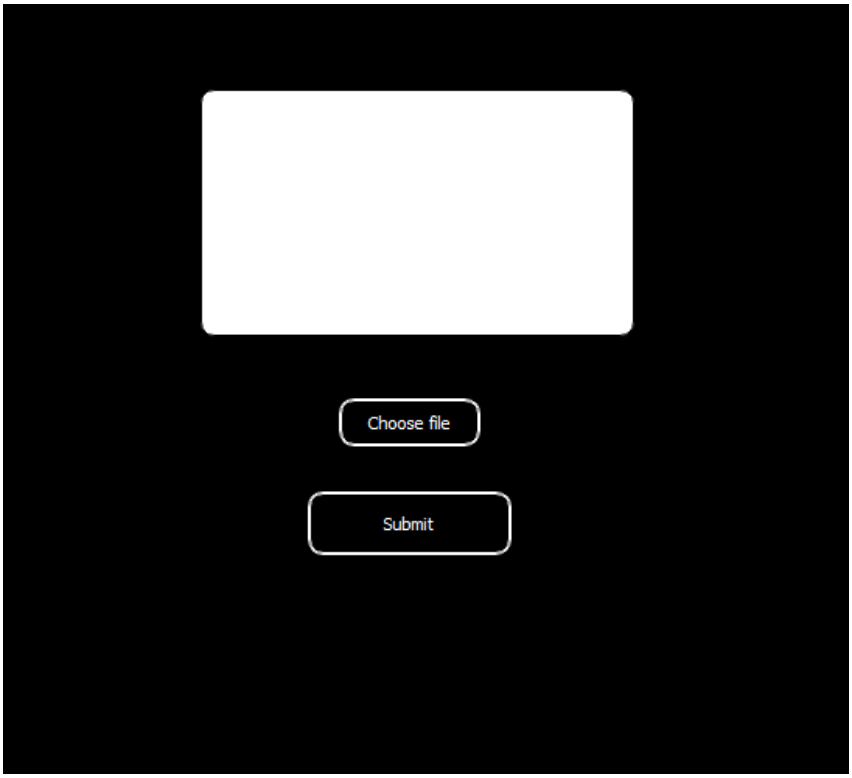


Figure 16 : Interface graphique

Le premier bouton « choose file » nous permet d'afficher la boîte de dialogue, pour ensuite sélectionner le fichier audio pour lequel on veut extraire son genre musical.

Le fichier audio sera donc chargé pour en extraire les données qui devront d'abord être normalisé et ensuite traité par notre modèle stocké dans un fichier de type h5, afin d'afficher le résultat, qui n'est autre que le genre de l'audio reconnu.

Conclusion :

Notre projet consistait à la réalisation d'une application permettant la classification des pistes audio selon le genre de musique en utilisant un réseau de neurones (backpropagation neural network) pour faire cette classification, à l'aide d'un ensemble de pistes audio libellés. Pour réaliser ce projet, on a commencé par une étude des variables que le modèle utilisera pour apprendre à faire la classification des genres de musique. Ensuite vient la partie de l'analyse des données extraites depuis l'ensemble des pistes audio, afin d'afficher les graphes et les tableaux qui nous donnerons plus d'informations sur l'ensemble de ses données. En troisième partie, viennent les architectures des modèles qu'on a adoptés pour ensuite choisir la plus optimale. Et enfin viens la partie de l'interface graphique qui a été développé à l'aide de l'outil PyQt5.