Université Mohammed V
Faculté des Sciences
Rabat

كلية العلوم الرباط
Faculté des sciences Rabat

MACHINE LEARNING

Projet : Classification des genres musicaux

Encadrer par Mme : MANAR ABOUREZQ

Travail Réaliser par : ELKHAZRAJE Othman

04 JUILLET 2021

MASTER Intelligent Processing Système FI

Table des matières

ABSTRACT :

La musique est aujourd'hui une partie importante du contenu Internet : le net est probablement la source la plus importante de morceaux de musique, avec plusieurs sites dédiés à la diffusion, la distribution et la commercialisation de la musique. Les entreprises utilisent aujourd'hui la classification musicale, soit pour pouvoir faire des recommandations à leurs clients (comme Spotify, Soundcloud) soit simplement en tant que produit (par exemple, Shazam).

De nos jours, la représentation du signal musical n'est plus analogue à l'onde sonore d'origine. Le signal analogique est échantillonné plusieurs fois par seconde et transformé par un convertisseur analogique-numérique en une séquence de valeurs numériques à une échelle appropriée. Cette séquence représente le signal audio numérique de la musique et peut être utilisée pour reproduire la musique.

Dans ce rapport, nous étudions l'impact des algorithmes d'apprentissage automatique dans le développement de modèles de classification automatique de la musique visant à capturer les distinctions de genres.

INTRODUCTION :

Un genre musical est une catégorisation de morceaux de musique qui partagent un certain style. La musique est également classée selon des critères non musicaux tels que l'origine géographique, bien qu'une seule région géographique comprennent normalement une grande variété de sous-genres. Tout genre ou sous-genre musical donné peut être défini par les instruments de musique utilisés, les techniques, les styles, le contexte ou les thèmes structurels.

Dans ce travail, nous avons choisir d'abord une collection d'exemples de genres largement reconnus et puis évaluer les performances de différents algorithmes d'apprentissage,

Nous allons essayer de modéliser un classificateur pour classer les chansons en différents genres. On va sélectionner des fichiers MP3 nommés au hasard sur notre disque dur, qui sont supposés contenir de la musique. Notre tâche est de les trier selon le genre musical dans différents dossiers tels que Blues, classique, country, pop, rock et métal.

Aussi on vous explique comment entraîner des modèles de classification automatique de musiques. Nous passerons par le nettoyage et la préparation des données d'entraînement. Nous entraînerons ensuite plusieurs modèles de classifications afin de pouvoir les comparer.

Les algorithmes de machine Learning :

Définition :

Le Machine Learning est une technologie d'intelligence artificielle permettant aux ordinateurs d'apprendre sans avoir été programmés explicitement à cet effet. Pour apprendre et se développer, les ordinateurs ont toutefois besoin de données à analyser et sur lesquelles s'entraîner.

Le développement d'un modèle de Machine Learning repose sur quatre étapes principales. En règle générale, c'est un Data Scientist qui gère et supervise ce procédé.

La première étape consiste à sélectionner et à préparer un ensemble de données d'entraînement. Ces données seront utilisées pour nourrir le modèle de Machine Learning pour apprendre à résoudre le problème pour lequel il est conçu. Les données peuvent être étiquetées, afin d'indiquer au modèle les caractéristiques qu'il devra identifier. Elles peuvent aussi être non étiquetées, et le modèle devra repérer et extraire les caractéristiques récurrentes de lui-même.

Dans les deux cas, les données doivent être soigneusement préparées, organisées et nettoyées. Dans le cas contraire, l'entraînement du modèle de Machine Learning risque d'être biaisé. Les résultats de ses futures prédictions seront directement impactés.

La deuxième étape consiste à sélectionner un algorithme à exécuter sur l'ensemble de données d'entraînement. Le type d'algorithme à utiliser dépend du type et du volume de données d'entraînement et du type de problème à résoudre.

La troisième étape est l'entraînement de l'algorithme. Il s'agit d'un processus itératif. Des variables sont exécutées à travers l'algorithme, et les résultats sont comparés avec ceux qu'il aurait du produire. Les " poids " et le biais peuvent ensuite être ajustés pour accroître la précision du résultat. On exécute ensuite de nouveau les variables jusqu'à ce que l'algorithme produise le résultat correct la plupart du temps. L'algorithme, ainsi entraîné, est le modèle de Machine Learning.

La quatrième et dernière étape est l'utilisation et l'amélioration du modèle. On utilise le modèle sur de nouvelles données, dont la provenance dépend du problème à résoudre.

Les différents types d'algorithmes de Machine Learning :

On distingue différents types d'algorithmes Machine Learning. Généralement, ils peuvent être répartis en deux catégories : supervisés et non supervisés.

Dans le cas de l'apprentissage supervisé, les données utilisées pour l'entraînement sont déjà '' étiquetées ''. Par conséquent, le modèle de Machine Learning sait déjà ce qu'elle doit chercher (motif, élément…) dans ces données. À la fin de l'apprentissage, le modèle ainsi entraîné sera capable de retrouver les mêmes éléments sur des données non étiquetées.

Parmi les algorithmes supervisés, on distingue les algorithmes de classification (prédictions non-numériques) et les algorithmes de régression (prédictions numérique). En fonction du problème à résoudre, on utilisera l'un de ces deux archétypes.

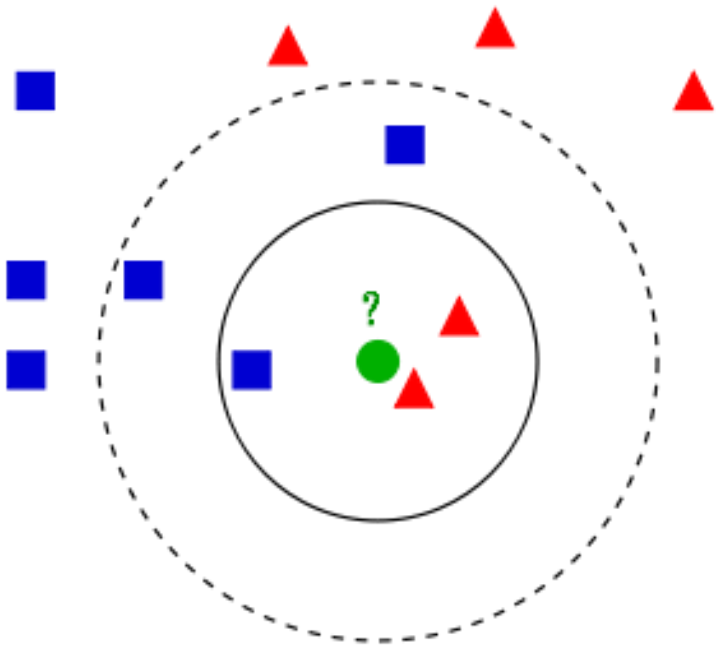
L'apprentissage non supervisé, au contraire, consiste à entraîner le modèle sur des données sans étiquettes. La machine parcourt les données sans aucun indice, et tente d'y découvrir des motifs ou des tendances récurrents. Cette approche est couramment utilisée dans certains domaines, comme la cybersécurité.

Parmi les modèles non-supervisés, on distingue les algorithmes de clustering (pour trouver des groupes d'objets similaires), d'association (pour trouver des liens entre des objets) et de réduction dimensionnelle (pour choisir ou extraire des caractéristiques).

KNN (K Nearst Neighbors) :

K-NN est un algorithme standard de classification qui repose exclusivement sur le choix de la métrique de classification. Il est "non paramétrique" (seul k doit être fixé) et se base uniquement sur les données d'entraînement.

L'idée est la suivante : à partir d'une base de données étiquetées, on peut estimer la classe d'une nouvelle donnée en regardant quelle est la classe majoritaire des k données voisines les plus proches (d'où le nom de l'algorithme). Le seul paramètre à fixer est k, le nombre de voisins à considérer (voir figure).



Random Forest :

Le random forest ou forêt d'arbres décisionnels est un algorithme crée en 1995 par HO, puis formellement proposé par les scientifiques Adele Cutler et Leo Breiman, en 2001. Il est particulièrement efficace en termes de prédictions dans le domaine du machine learning, du deep learning et de l'intelligence artificielle (IA).

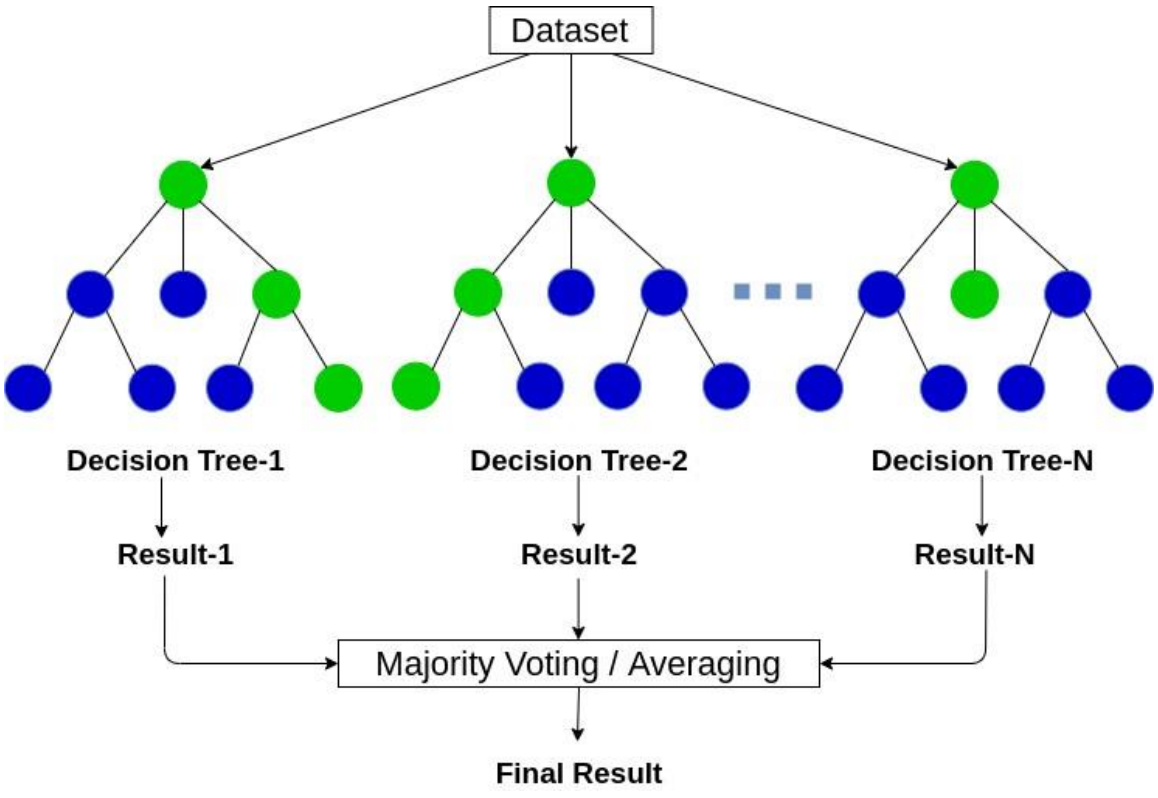
Le random forest est composé de plusieurs arbres de décision, travaillant de manière indépendante sur une vision d'un problème. Chacun produit une estimation, et c'est l'assemblage des arbres de décision et de leurs analyses, qui va donner une estimation globale. En somme, il s'agit de s'inspirer de différents avis, traitant un même problème, pour mieux l'appréhender. Chaque modèle est distribué de façon aléatoire aux sousensembles d'arbres décisionnels.

Le random forest est un modèle d'apprentissage, dont l'efficacité dépend fortement de la qualité de l'échantillon de données de départ.

- Un random forest de classification :

Un random forest de classification est également basé sur le système du bagging.

Par contre, l'estimation finale se réalise à partir d'une méthode de classification. On choisit la catégorie de réponse la plus fréquente. Plutôt qu'utiliser tous les résultats obtenus, on procède à une sélection en recherchant la prévision qui revient le plus souvent.



SVM (machines à vecteurs de support) :

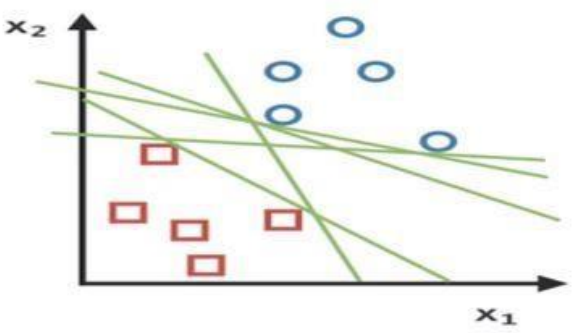
Un SVM qui est l'acronyme de Support Vector Machines, soit machines à vecteurs support en français, parfois traduit par séparateur à vaste marge pour garder l'acronyme, c'est un algorithme d'apprentissage automatique, et qui est très efficace dans les problèmes de classification, cette classification qui va nous
Permet de trouver l'hyperplan optimal qui maximise la margine en deux classes.

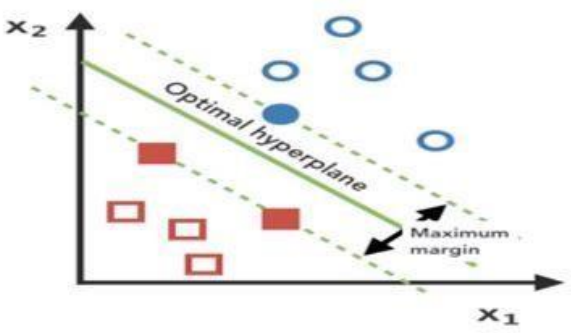
Comment fonction l'algorithme :

L'algorithme de SVM a pour objectif de trouver la séparation entre deux classes d'objets avec l'idée que plus la séparation est large, plus la classification est robuste. Dans sa forme la plus simple, celle d'une séparation linéaire et de classes séparables, l'algorithme sélectionne l'hyperplan qui sépare le jeu d'observations en deux classes distinctes de façon à maximiser la distance entre l'hyperplan et les observations les plus proches de l'échantillon d'apprentissage.

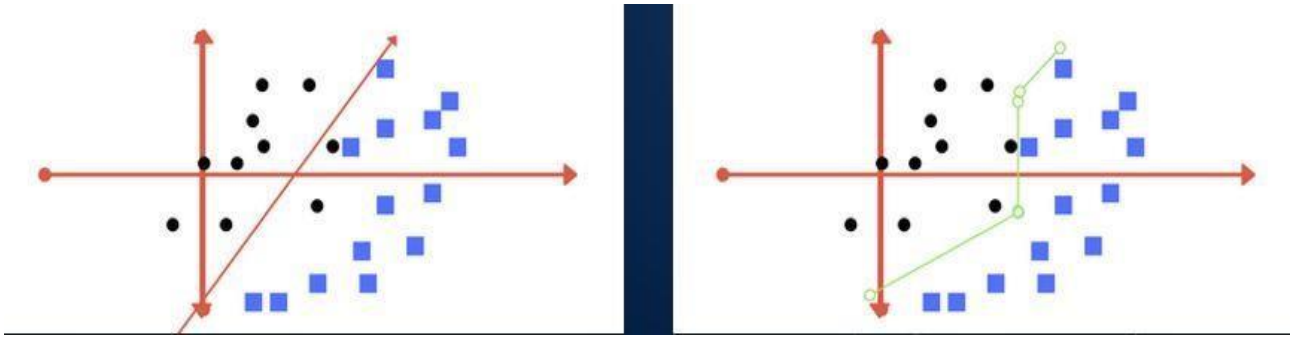
- Premièrement on observe le nombre de classes, et on cherche la ligne optimal(hyperplane) pour le dessiner (qui doit être entre les points les plus proches et qui sont dans des différentes classes).
- Les points (ou les vecteurs si notre dim > 2) s'appels **des (support vectors).**
-
-

- On dit que l'**hyperplan** est optimal, si les distances entre lui et les support vectors sont maximisé, dans ce cas cette distance s'appelle **(la Marge).**
- Apres l'algorithme va nous proposer les éléments de chaque classe en utilisant l'Hyperplan comme une frontière de décision.





Des fois on trouve qu'un élément d'une classe peut être très proche de l'autre classe ce qui va diminuer l'efficacité de notre modèle, pour traiter ce problème l'hyperplan va être la médiane, et après on peut choisir soit qu'il ignore toutes les exceptions, ou adapter l'hyperplan pour les inclure.

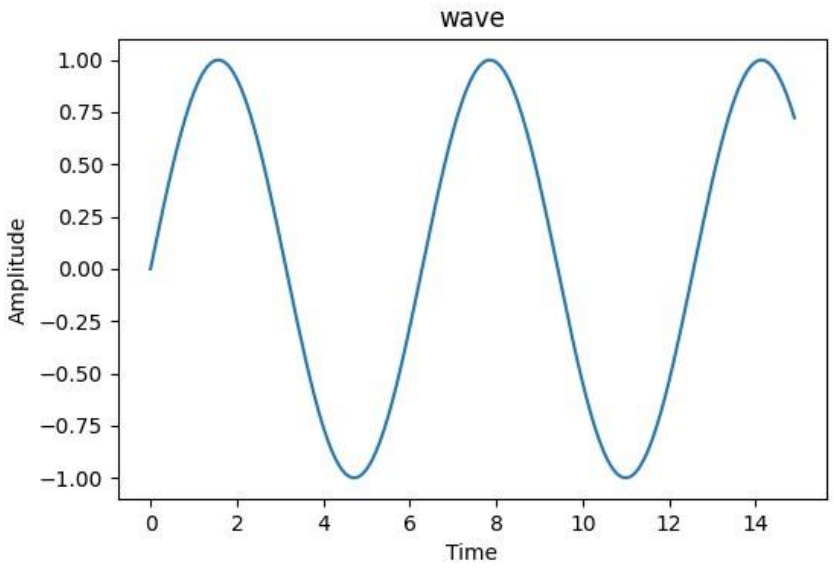


Audio processing :

Fondamentaux

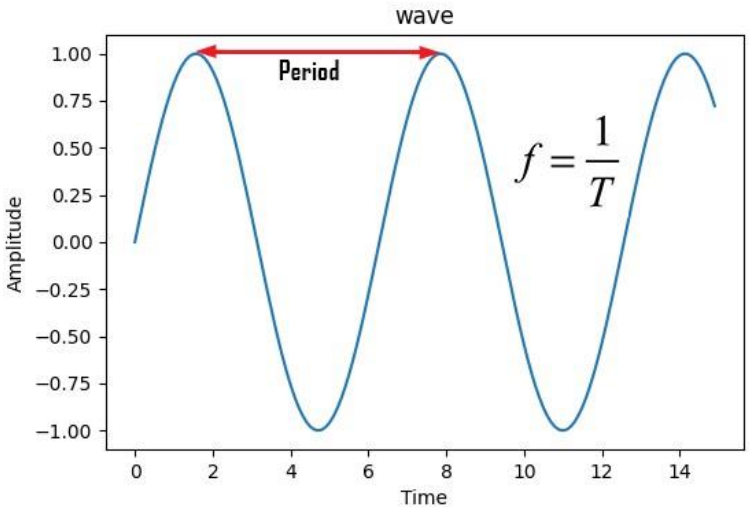
Le Son

Le **son** est une vibration qui se propage comme une onde acoustique, à travers un milieu de transmission qu'en peut représenter sous la forme d'un courbe comme la figure suivante

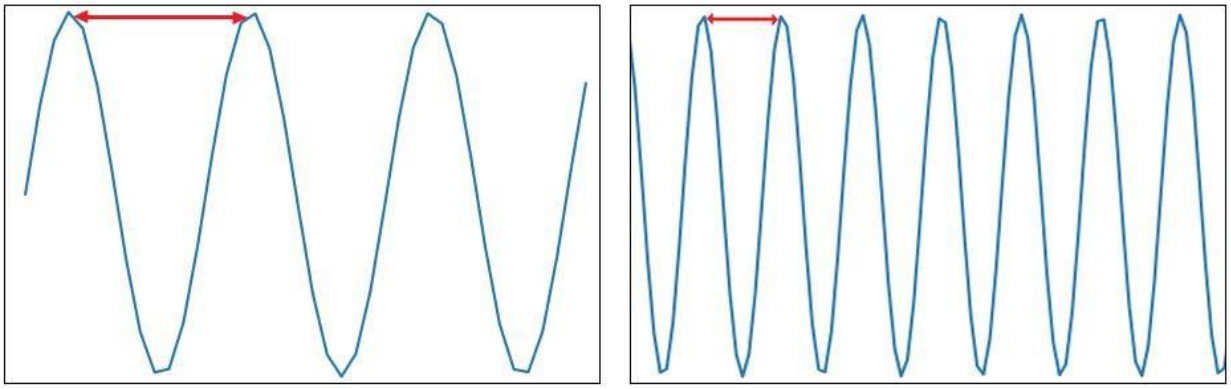


Ce courbe est la quelle que caractéristiques

- **La Fréquence**

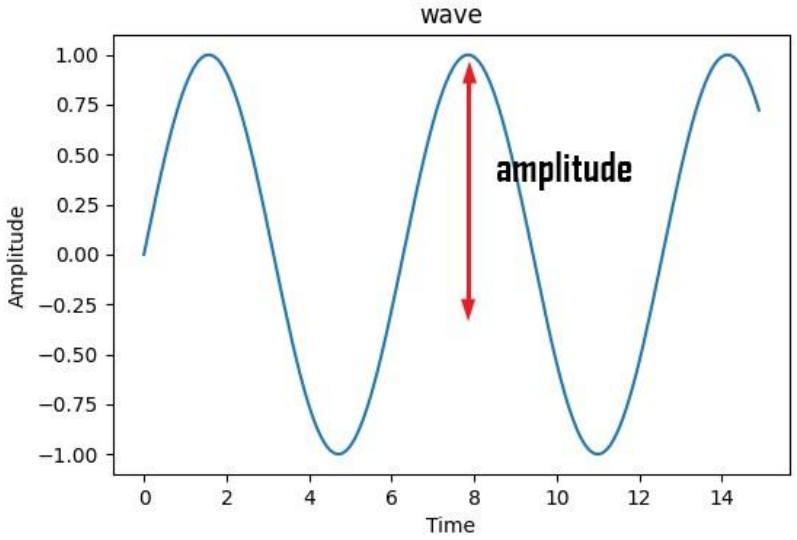


- La période et la distance entre deux maxima successifs.
- La fréquence est le nombre de fois que la période se reproduit par unité de temps
- La fréquence f est l'inverse de la période
- Plus la période est basse, plus la fréquence est élevée (vice versa).



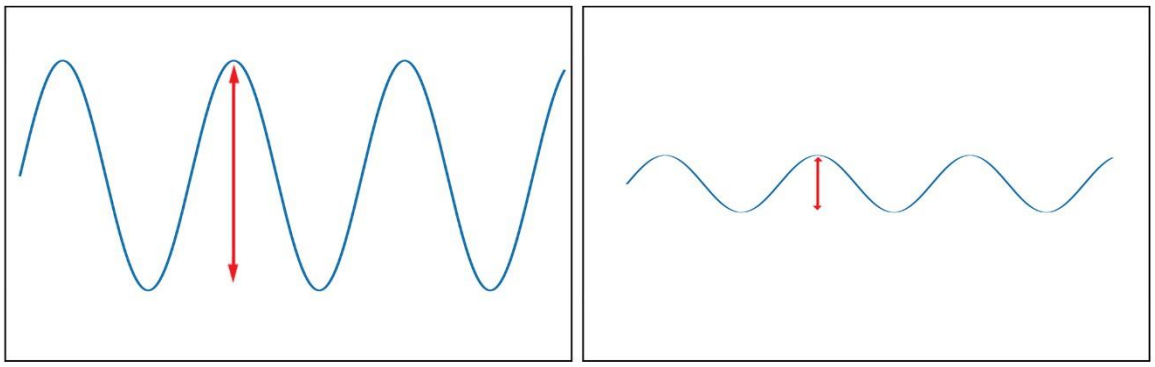
On peut conclure depuis la valeur de la fréquence Le Pitch de l'audio.
Le Pitch d'un son est à quel point le son est haut ou bas. Un son aigu a un Pitch élevé et un son grave a un Pitch faible.

- **Amplitude et l'intensité**



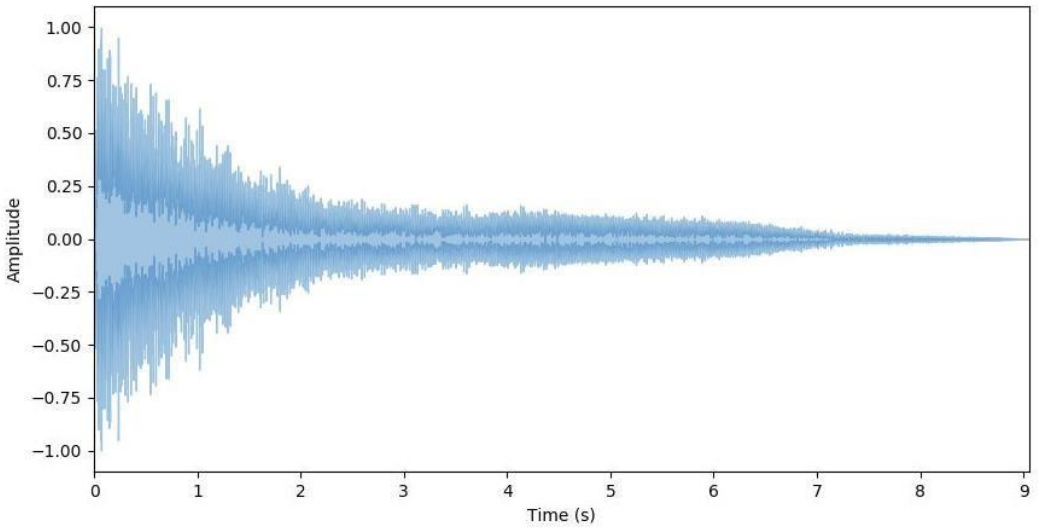
-

L'intensité d'un son est liée à l'amplitude.
L'amplitude est plus grande dans un son fort que dans un son faible



1. Les techniques d'extraction des caractéristiques d'une piste audio

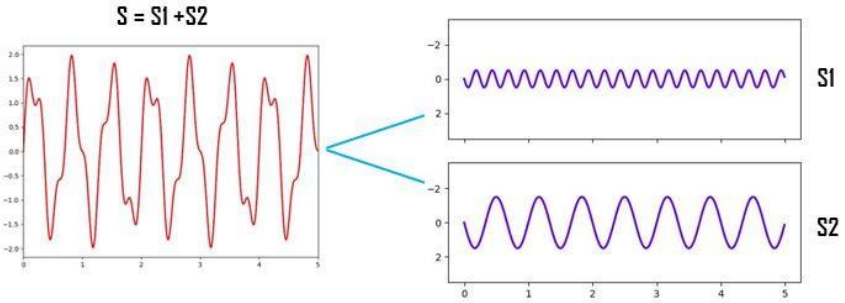
Pour mieux comprend nous utilise une vraie onde sonore d'une touche de piano pour décrire les étapes que nous avons suivre pour prépare notre dataset.



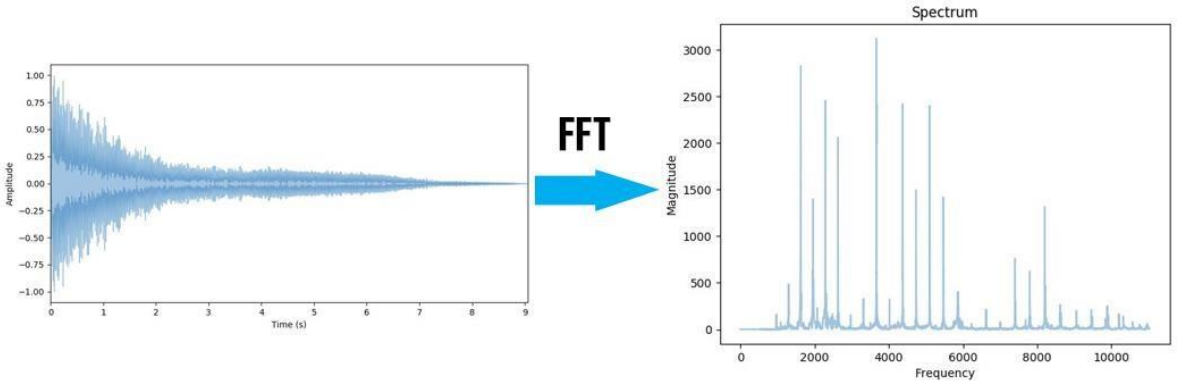
En remarque du le premier instant que notre courbe est plus compliqué par rapport au coures précèdent. En peur voir que notre courbe n'apporte aucune valeureux information dans leur format actuel qui peux nous aide à classifier nos sons de musique.
Pour extraire des informations plus valeureuses de notre data nous avons utilisé Transformée de Fourier.

- Fourier transforme

Décomposer un son périodique complexe en somme d'ondes sinusoïdales oscillant à différentes fréquences.



Dans audio processing nous utilisons FT pour transformer la courbe d'une représentation par rapport au temps a une représentation par rapport au fréquence.



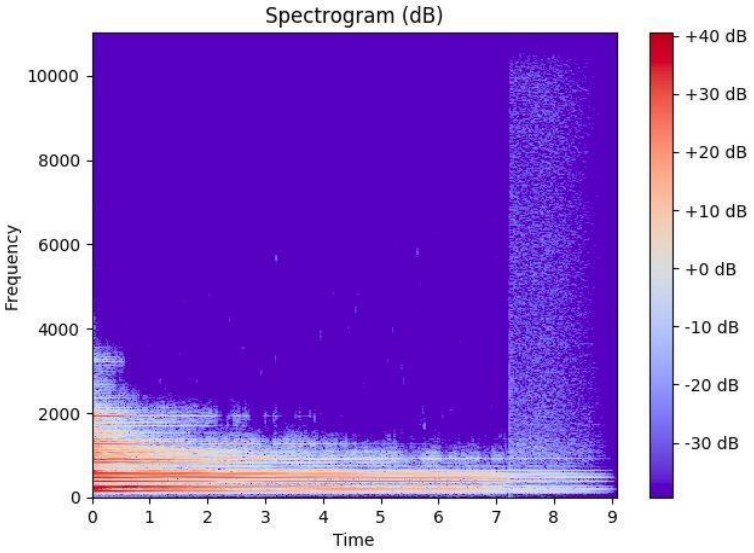
Dans la figure au haut le FFT (Fast Fourier transforme) dons un globale représentations des **fréquences** on fonction de la **magnitude** dans tout la piste audio mais nous avons perdu une l représentation de temps qu'il a une trais grande importance parce que l'onde sonore est une série temporelle.

Pour résolu ce problème on a utilisé le **Short Time Fourier Transforme (STFT)**

Short Time Fourier Transform (STFT)

Le STFT méthode consiste à couper l'onde sonore a plusieurs intervalles et après elle applique lui-même le FFT sur chacun des intervalles pour finalement elle nous donne un **spectrogramme** qui représente la **fréquence** le **temps** et la **magnitude**.

L'intervalle est un ensemble de frame que nous précisent avent l'utilisation de STFT



Mel Frequency Cepstral Coefficients (MFCCs)

C'est un concept avance dans le domaine de digital audio processing. Il est capable de Capturez les aspects texturaux du son et approximé le système auditif humain.

- Description des attributs de DataSet

-

Après l'explication des différents méthodes et technique de trament d'extraction de donnes depuis les signaux audio. En peut maintenant comprend la source de chacun des attribué de notre dataset.

- **Length** : la dure de la piste audio
- **Chroma_stft_mean & var** : la moyenne et la variance
- **rms_mean & var** : Moyenne et la variance quadratique
- **Spectral_centroid_mean & var** : Moyenne et la variance de spectrogramme de FTT
- **harmony_mean & var** : Moyenne et la variance de l'harmonie
- **MFCCs_ mean & var** : Moyenne et la variance de chaque intervalle

Dataset et préparation de l'environnement de travail:

Pour ce projet nous utiliserons la base **GTZAN**. Elle comporte 1000 pistes de 30 secondes chacune, réparties suivant 10 classes différentes :

- Blues
- Classic
- Country
- Disco
- Hip-Hop
- Jazz
- Métal
- Pop
- Reggae

- Rock

  - Commençons tout d'abord par importer toutes les packages qu'on nécessite :

```
import matplotlib.pyplot as plt

import pandas as pd
import os
import numpy as np
import seaborn as sns

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score,classification_report
from sklearn import preprocessing
from sklearn.model_selection import train_test_split,validation_curve
```

  - On importe notre dataset par pandas 'pd.read_csv' et on affiche les cinq premières lignes avec la fonction head() :

```
df = pd.read_csv(path+'/features_3_sec.csv')
df.head()
```

Python

	filename	length	chroma_stft_mean	...	mfcc20_mean	mfcc20_var	label
0	blues.00000.0.wav	66149	0.335406	...	-0.243027	43.771767	blues
1	blues.00000.1.wav	66149	0.343065	...	5.784063	59.943081	blues
2	blues.00000.2.wav	66149	0.346815	...	2.517375	33.105122	blues
3	blues.00000.3.wav	66149	0.363639	...	3.630866	32.023678	blues
4	blues.00000.4.wav	66149	0.335579	...	0.536961	29.146694	blues

[5 rows x 60 columns]

  - Afficher le nombre de lignes et colonnes par la fonction shape :

```
df.shape
```

Python

```
(9990, 60)
```

  - Afficher une résumé données statistiques du DataFrame :

```
df.describe()
```

Python

	length	chroma_stft_mean	...	mfcc20_mean	mfcc20_var
count	9990.0	9990.000000	...	9990.000000	9990.000000
mean	66149.0	0.379534	...	-0.917584	57.322614
std	0.0	0.090466	...	5.253243	46.444212
min	66149.0	0.107108	...	-35.640659	0.282131
25%	66149.0	0.315698	...	-4.004475	30.011365
50%	66149.0	0.384741	...	-1.030939	44.332155
75%	66149.0	0.442443	...	2.216603	68.210421
max	66149.0	0.749481	...	34.212101	910.473206

[8 rows x 58 columns]

- Afficher des informations sur les données , le type d'index et les colonnes, les valeurs non NULL:

```
df.info()
```

Press Ctrl+Alt+Enter to execute cell

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9990 entries, 0 to 9989
Data columns (total 60 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   filename                 9990 non-null   object
 1   length                   9990 non-null   int64
 2   chroma_stft_mean         9990 non-null   float64
 3   chroma_stft_var          9990 non-null   float64
 4   rms_mean                 9990 non-null   float64
 5   rms_var                  9990 non-null   float64
 6   spectral_centroid_mean   9990 non-null   float64
 7   spectral_centroid_var    9990 non-null   float64
 8   spectral_bandwidth_mean  9990 non-null   float64
 9   spectral_bandwidth_var   9990 non-null   float64
 10  rolloff_mean             9990 non-null   float64
 11  rolloff_var              9990 non-null   float64
 12  zero_crossing_rate_mean  9990 non-null   float64
 13  zero_crossing_rate_var   9990 non-null   float64
 14  harmony_mean             9990 non-null   float64
```

- La somme pour chaque label :

```
df['label'].value_counts()
```

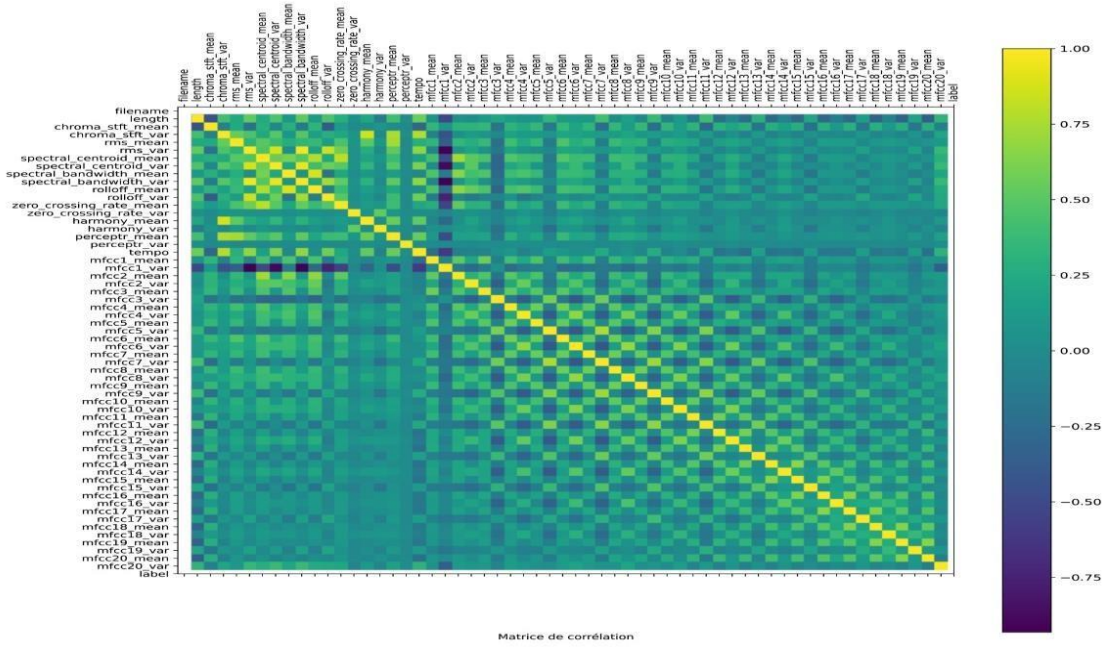
```
jazz         1000
pop          1000
metal        1000
reggae       1000
blues        1000
disco         999
classical     998
hiphop        998
rock          998
country       997
Name: label, dtype: int64
```

Corrélations :

Pour les corrélations on peut construire une matrice pour visualiser les corrélations :

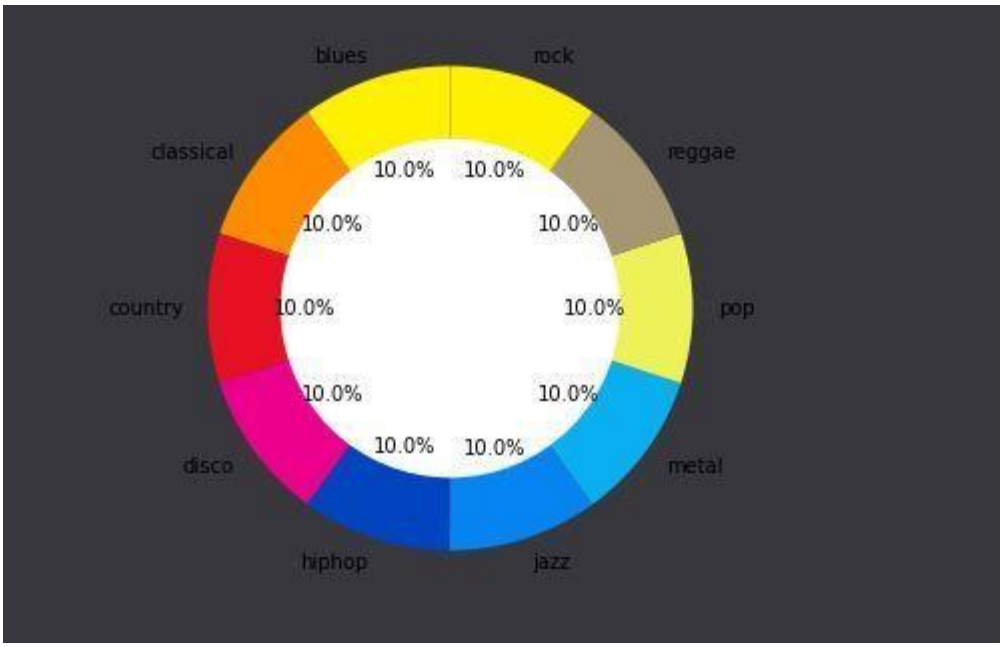
```
f = plt.figure(figsize=(14, 14))

plt.matshow(df.corr(), fignum=f.number)
plt.xticks(range(df.shape[1]), df.columns, fontsize=10, rotation=90)
plt.yticks(range(df.shape[1]), df.columns, fontsize=10)

cb = plt.colorbar()
cb.ax.tick_params(labelsize=11)
plt.title('Matrice de corrélation', fontsize=10, y=-0.15)
```

Le résultat obtenu :



On remarque de cette matrice de corrélation que lorsque la couleur est proche de la couleur jaune plus les variables sont corrélées positivement. Et plus la couleur se rapproche du bleu et plus les variables sont corrélées négativement. Il est clair que plusieurs variables sont fortement corrélées.

Visualisation des données :



Construction des modèles :

Dans cette partie nous allons construire des modèles de classification et allons les comparer pour voir lequel convient le mieux.

Voici les différentes méthodes qu'on a utilisé :

- Knn
- Random forest
- SVM

Normalisation des données :

La normalisation fait référence à la remise à l'échelle des attributs numériques à valeur réelle dans une plage de 0 à 1.

La normalisation des données est utilisée dans l'apprentissage automatique pour rendre la formation des modèles moins sensible à l'échelle des fonctionnalités. Cela permet à notre modèle de converger vers de meilleurs poids et, à son tour, conduit à un modèle plus précis.

La normalisation rend les caractéristiques plus cohérentes les unes avec les autres, ce qui permet au modèle de prédire les sorties avec plus de précision.

- **Code python :**

Python fournit la bibliothèque de *Preprocessing*, qui contient la fonction *normalize* pour normaliser les données.

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['label'] = label_encoder.fit_transform(df['label'])
```

```
cols = X.columns
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(X)
X = pd.DataFrame(np_scaled, columns = cols)
X
```

	length	chroma_stft_mean	...	mfcc20_mean	mfcc20_var
0	0.0	0.418094	...	0.568806	0.043294
1	0.0	0.345804	...	0.619547	0.066839
2	0.0	0.467927	...	0.456491	0.040426
3	0.0	0.207811	...	0.475485	0.124017
4	0.0	0.541335	...	0.502658	0.063868
...	...	...	...	...	...
9985	0.0	0.388220	...	0.505681	0.073662
9986	0.0	0.463454	...	0.533065	0.054319
9987	0.0	0.282028	...	0.621734	0.021704
9988	0.0	0.481854	...	0.615654	0.047280
9989	0.0	0.331233	...	0.514927	0.064752

[9990 rows x 58 columns]

k-nn (k nearest neighbours) :

On commence par reconstruire nos ensembles d'entraînement, et on divise le dataset entre train data et test data.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=111)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((6993, 58), (2997, 58), (6993,), (2997,))
```

Entraînement du modèle :

```
knn = KNeighborsClassifier(n_neighbors=19)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=19, p=2,
                     weights='uniform')
```

```
print('Train score : ', knn.score(X_train,y_train))
print('Test score : ', knn.score(X_test,y_test))
```

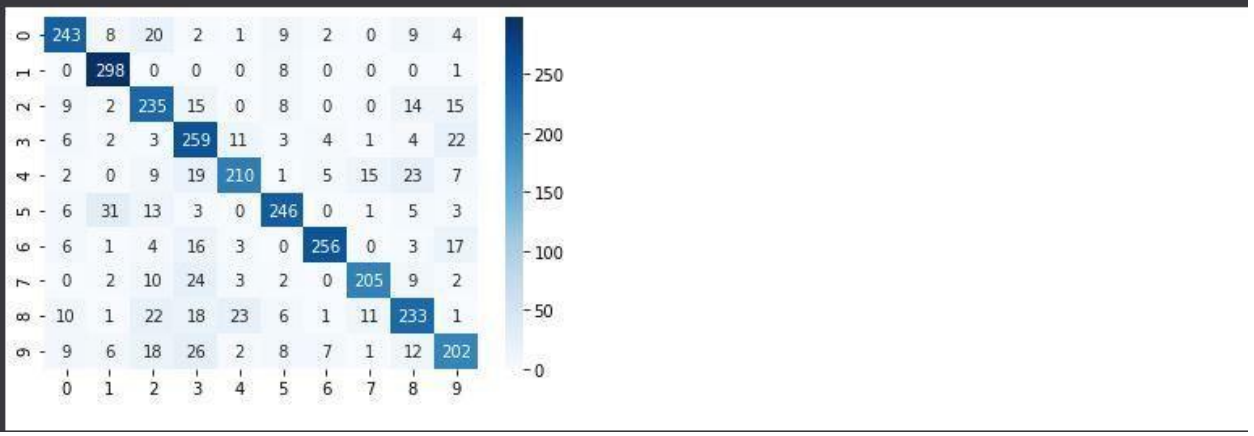
 Voilà le résultat obtenu :

```
Train score :  0.8411268411268411
Test score :  0.7964631297964632
```

 Confusion matrix :

```
preds = knn.predict(X_test)
c_matrix = confusion_matrix(y_test, preds)
print('Accuracy:', round(accuracy_score(y_test, preds), 5), '\n')
sns.heatmap(c_matrix, annot=True,fmt='1', cmap='Blues')
```

```
Accuracy: 0.79646


<matplotlib.axes._subplots.AxesSubplot at 0x7f6882d44350>
```



 Les résultats sont un peu bien avec K-nn, notamment pour certaines classes comme la musique (classique : 1), (disco : 3) et (metal : 6). Cela peut s'expliquer par le fait que ce sont des styles musicaux plus facilement détectables

SVM (support vector machine) :

Entraînement du modèle :

```
svm = SVC(decision_function_shape="ovo")
svm.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
print('Train score : ', svm.score(X_train,y_train)) print('Test score
: ', svm.score(X_test,y_test))
```

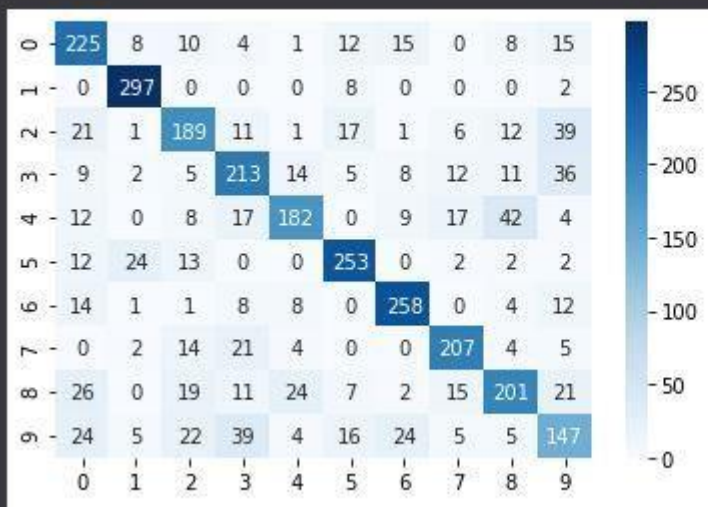
 voila le résultat obtenu :

```
Train score :  0.7772057772057772
Test score :  0.7247247247247247
```

 Confusion matrix :

```
preds = svm.predict(X_test) c_matrix =
confusion_matrix(y_test, preds) sns.heatmap(c_matrix,
annot=True,fmt='1', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6882cddfd0>
```



```
print(classification_report(preds, y_test))
```

```
              precision    recall  f1-score   support

           0       0.76      0.66      0.70       343
           1       0.97      0.87      0.92       340
           2       0.63      0.67      0.65       281
           3       0.68      0.66      0.67       324
           4       0.63      0.76      0.69       238
           5       0.82      0.80      0.81       318
           6       0.84      0.81      0.83       317
           7       0.81      0.78      0.79       264
           8       0.62      0.70      0.65       289
           9       0.51      0.52      0.51       283

    accuracy                           0.72      2997
   macro avg       0.73      0.72      0.72      2997
weighted avg       0.73      0.72      0.73      2997
```

Random Forest :

SVM et k-nn ne sont pas très adaptés à ce genre de données, c'est ce qui explique les performances médiocres que l'on obtient.

Pour des dataset comme celui que l'on a construits, il est préférable d'utiliser des méthodes qui reposent sur des arbres de décisions. Random forest ou XGBoost sont de très bons choix !

Entrainement du modèle :

```
randomforest = RandomForestClassifier(n_estimators=1000, max_depth=None, random_state=21)
randomforest.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=21, verbose=0,
                       warm_start=False)
```

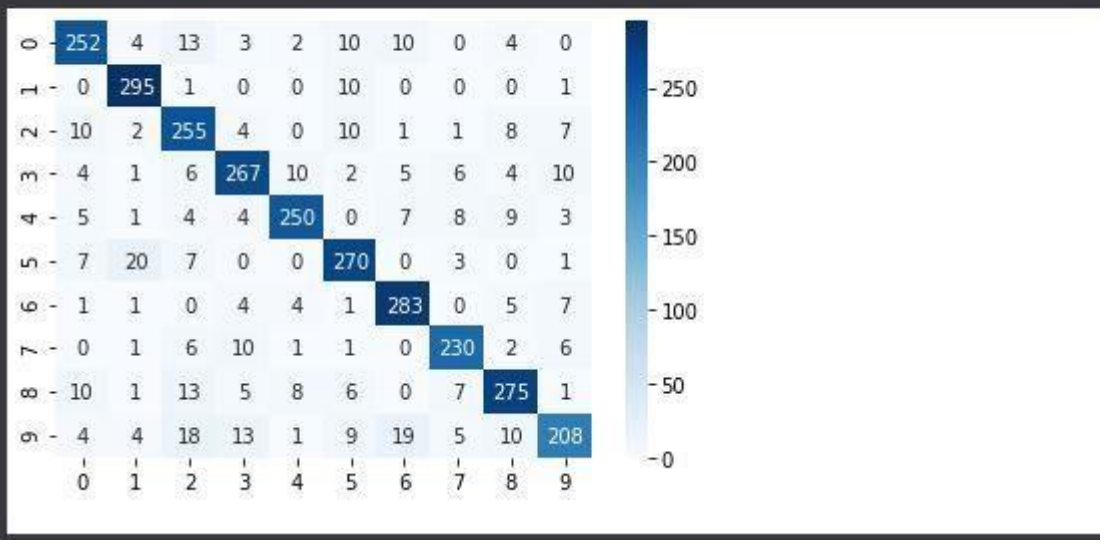
```
preds = randomforest.predict(X test)
print('Accuracy:', round(accuracy score(y test, preds), 5), '\n')
```

```
Accuracy: 0.86253
```

Matrice de confusion :

```
preds = svm.predict(X_test) c_matrix =
confusion_matrix(y_test, preds) sns.heatmap(c_matrix,
annot=True,fmt='1', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68822da2d0>
```



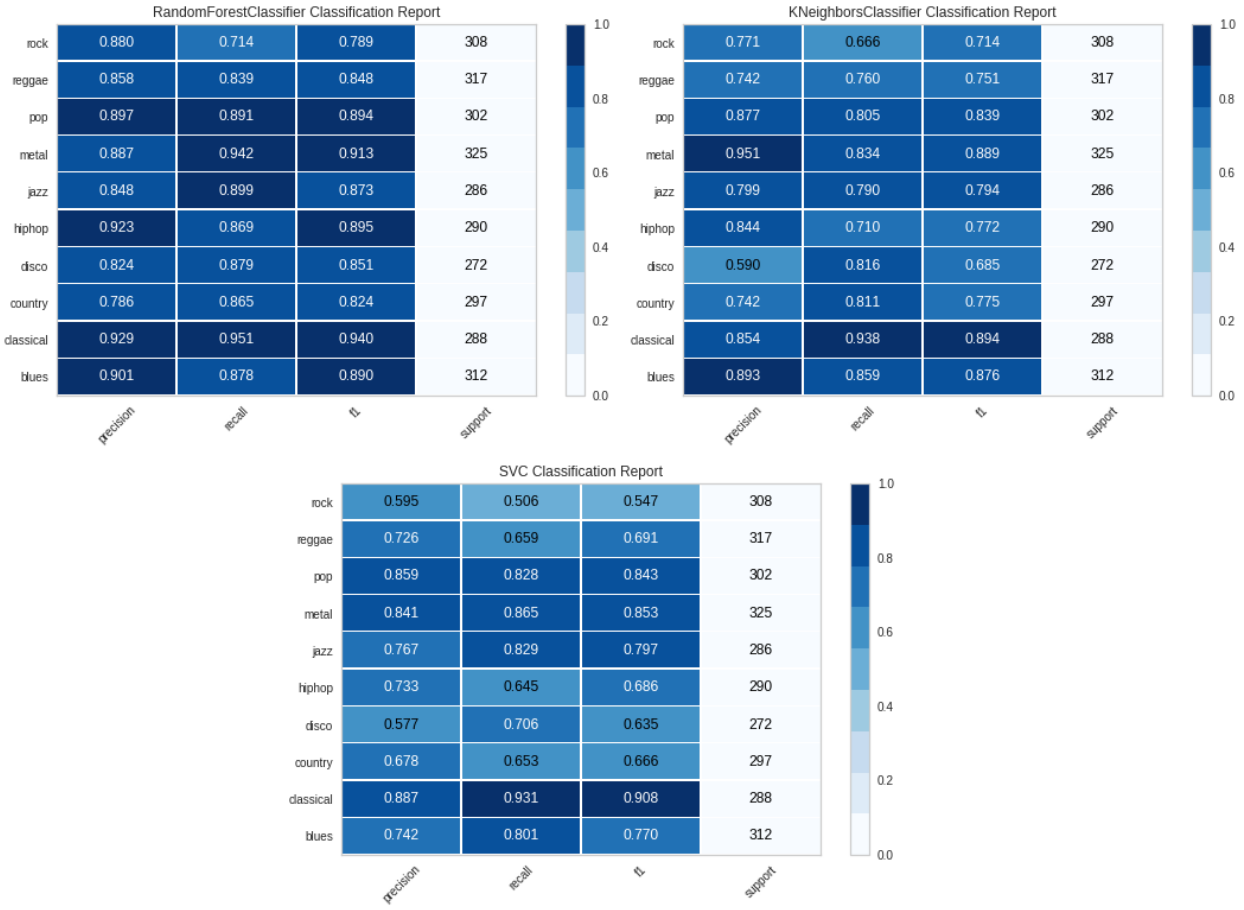
```
zero_one_loss error : 412
accuracy_score on test dataset : 0.8625291958625292
              precision    recall  f1-score   support

           0       0.85      0.86      0.85       293
           1       0.96      0.89      0.93       330
           2       0.86      0.79      0.82       323
           3       0.85      0.86      0.85       310
           4       0.86      0.91      0.88       276
           5       0.88      0.85      0.86       319
           6       0.92      0.87      0.90       325
           7       0.89      0.88      0.89       260
           8       0.84      0.87      0.86       317
           9       0.71      0.85      0.78       244

    accuracy                           0.86      2997
```

Evaluation de la classification des modèles

1)-Rapport de classification

Le rapport de classification fournit les principales mesures de classification par classe.

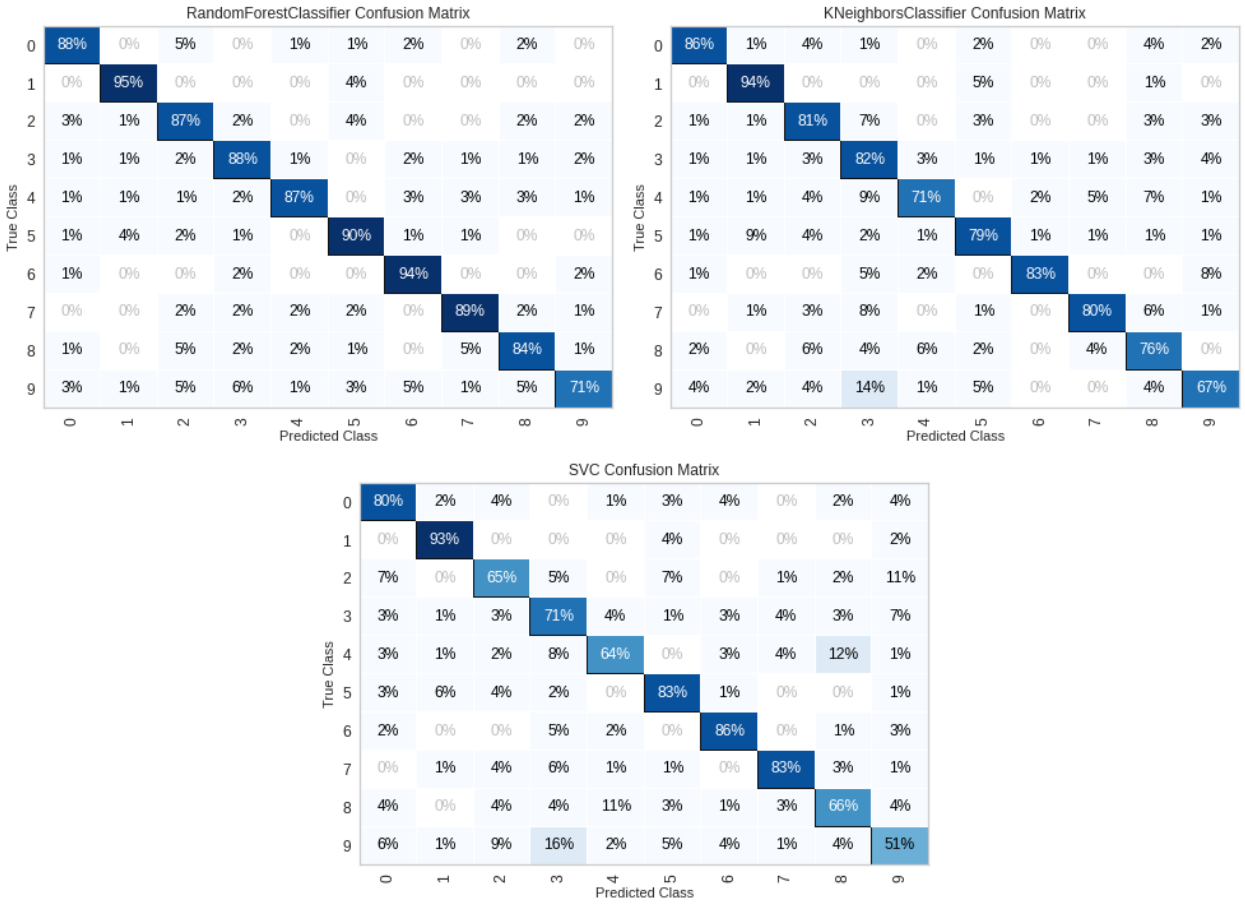
- Précision : Proportion d'observations réellement positives parmi toutes les observations que le modèle a prédites comme positives
- Rappel (Recall) : Proportion d'observations positives que le modèle a correctement prédites, i.e. taux de vrais positif
- F1 score : est une moyenne harmonique pondérée de précision et de rappel normalisée entre 0 et 1. Le score F de 1 indique un équilibre parfait car la précision et le rappel sont inversement liés. Un score F1 élevé est utile lorsqu'un rappel et une précision élevés sont importants.



En peut remarquer que le modèle de Random Forest il a un très bon compromis entre la précision le rappel (Recall) dans tous les classes ce qui traduit dans la valeur de score F1.
Pour le modèle KNN il donne des résultats plus ou moins bon, mais le SVM il a échouer dans la plupart des classes.

Matrice de confusion

La matrice de confusion est parmi les piliers de domaine d'évaluation des modèles elle permet de calculer plusieurs mesures comme la précision le rappel (Recall) et le F1 score. Une matrice de confusion montre la combinaison des classes réelles et prévues. Chaque ligne de la matrice représente les instances d'une classe prédite, tandis que chaque colonne représente les instances d'une classe réelle.



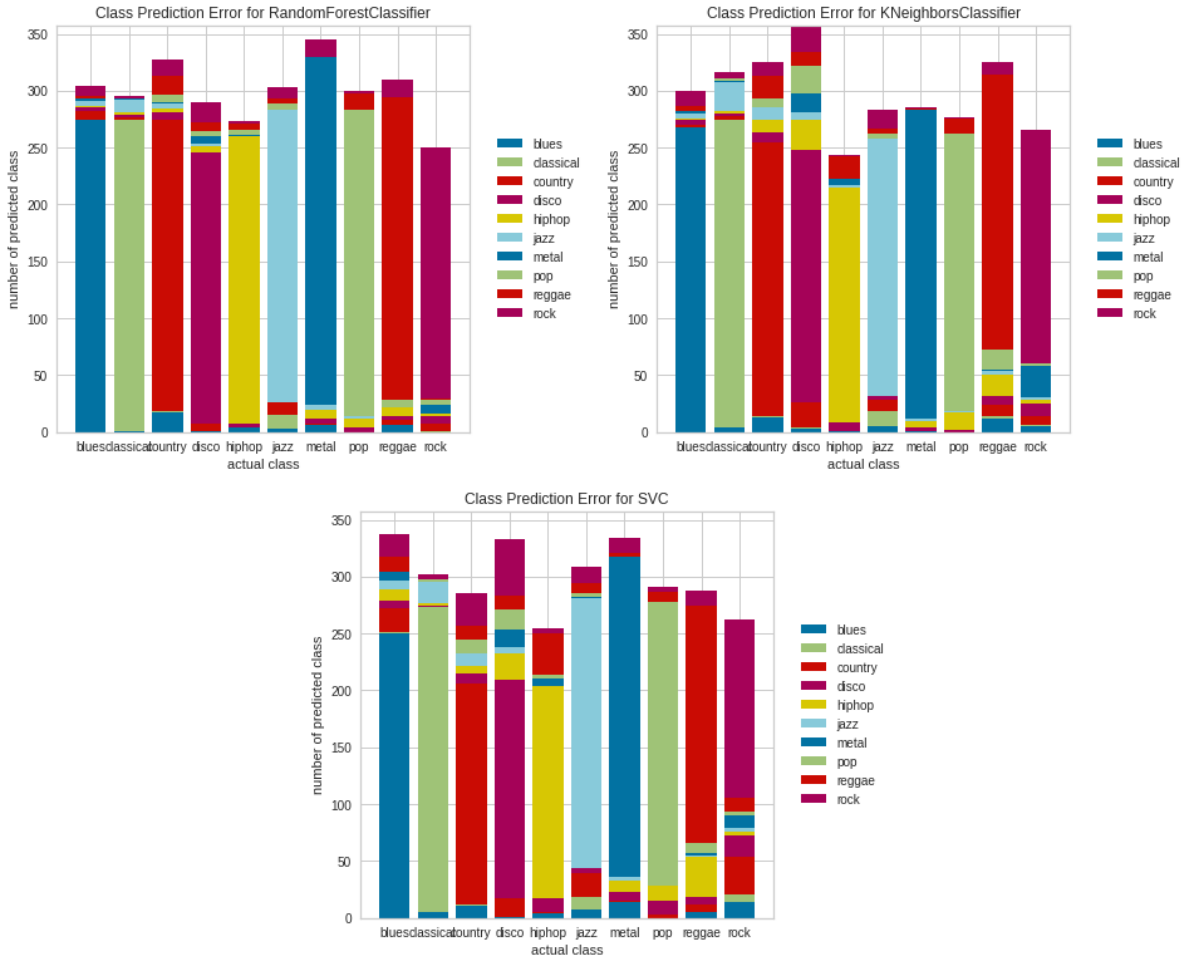
Nous avons déjà utilisé la matrice dans le précédant chapitre mais maintenant pour facilite la lecture et la comparaison entre les matrices nous avons exprimés les valeurs de la matrice par des pourcentages

La première chose quand peut remarquer les pourcentages dans la diagonal des matrices sont vraiment élevée par rapport à le reste ce qui indique que les trois modelés sont en génal bien classer notre dataset.

Le modèle de Random Forest est toujours donne des très bons résultats jusqu'à maintenant par contre le SVM a trompé dans plusieurs classes de notre dataset par exemple la classe du type musical Rock (label numéro 9) il a bien classé juste la moitié par rapport aux autres modelés [KNN 67%, Random Forest 71%] et en peut remarquer ça clairement dans les plots d'extension **Class Prediction Error.**

Class Prédiction Error

Il s'agit d'une extension utile de la matrice de confusion et visualise les classes mal classées sous la forme d'une barre empilée. Chaque barre est une mesure composite des classes prédites.



Cette méthode aide à comprendre la distribution des valeurs des donnes bien classer. Par exemple presque la moitié des valeurs de qui sont classé du genre musical rock de modèle SVM sont mal classées.

Accuracy :

Model	Accuracy
Random forest	87.22%
k-nearest neighbors	79.84%
Support vector machine	74.2%

Université Mohammed V
Faculté des Sciences
Rabat

كلية العلوم الرباط
Faculté des sciences Rabat

CONCLUSION

Dans ce projet de classification des genres musicaux,
on a expliqué comment entraîner des modèles de classification
automatique de musiques sur les fichiers audio pour prédire son genre.
Nous travaillons à travers ce projet sur l'ensemble de données
de classification des genres musicaux GTZAN.
À l'avenir, nous espérons expérimenter d'autres types de méthodes
du Deep learning, étant donné qu'elles sont les plus
performantes. De plus, nous soupçonner que nous pouvons avoir
des opportunités de transfert d'apprentissage, par exemple
en classant la musique par artiste.