Claire DELGOVE
Nicolas GAUFFIN

FISE 2022
SDATA

RAPPORT

Classification supervisée, régression, et analyse de jeux de données issus de l'application Spotify®

Introduction

L'objectif de ce projet est d'étudier différents algorithmes d'analyse de données afin de comprendre leurs mécanismes. L'exercice 1 permet d'aborder les classifieurs, en effectuant des prédictions de genres musicaux. Le deuxième exercice permet de se pencher sur les algorithmes de régression en prédisant les popularités d'un grand nombre de chansons. Ce rapport est joint d'un fichier python dans lequel se trouve le code associé.

Exercice 1

Principe de l'exercice

Le but de l'exercice est de proposer une méthode de classification optimale pour prédire le genre musical de musiques disponibles dans la base de données de Spotify®. La performance des algorithmes testés sera quantifiée selon le F1_score.

Analyse du jeu de données d'entraînement

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31728 entries, 0 to 31727
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   danceability      31728 non-null  float64
 1   energy            31728 non-null  float64
 2   key               31728 non-null  int64
 3   loudness          31728 non-null  float64
 4   mode              31728 non-null  int64
 5   speechiness       31728 non-null  float64
 6   acousticness      31728 non-null  float64
 7   instrumentalness  31728 non-null  float64
 8   liveness          31728 non-null  float64
 9   valence           31728 non-null  float64
 10  tempo             31728 non-null  float64
 11  type              31728 non-null  object
 12  id                31728 non-null  object
 13  uri               31728 non-null  object
 14  track_href        31728 non-null  object
 15  analysis_url      31728 non-null  object
 16  duration_ms       31728 non-null  int64
 17  time_signature    31728 non-null  int64
 18  song_name         16103 non-null  object
 19  genre             31728 non-null  object
dtypes: float64(9), int64(4), object(7)
memory usage: 4.8+ MB
```

Le jeu de données d'entraînement se présente sous la forme d'un DataFrame composé d'environ 32 000 musiques caractérisées selon 20 attributs. Le 20e (**genre**) est celui que nous cherchons à prédire. Parmi les 19 autres, nous choisissons d'éliminer ceux qui n'apportent rien dans la caractérisation pure de la chanson (**type**, **uri**, **id**, **analysis_url**, **track_href**, **song_name**). Le jeu de données "efficace" contient alors 13 catégories et le genre musical, qui sont toutes de type float ou int. Aucune valeur manquante n'est présente, ce qui facilite le traitement.

Traitement préliminaire des données

On définit **x** le tableau de vecteurs composé des 13 attributs, et **y** le celui du Genre musical.

```
y = Train.genre
x = Train.drop('genre', axis=1)
```

Pour augmenter l'efficacité des algorithmes de classification, on choisit de réduire et centrer les données de **x** en utilisant la fonction **StandardScaler()** du module preprocessing de la bibliothèque Scikit_learn.

```
sc = StandardScaler()
x = sc.fit_transform(x)
```

Tous les tests ont été effectués sur le jeu d'entraînement avec le même mélange des données : la proportion pour le train a été fixée à 80%, et 20% pour le test. De plus, le mélange a été réalisé selon le même **Random_State**.

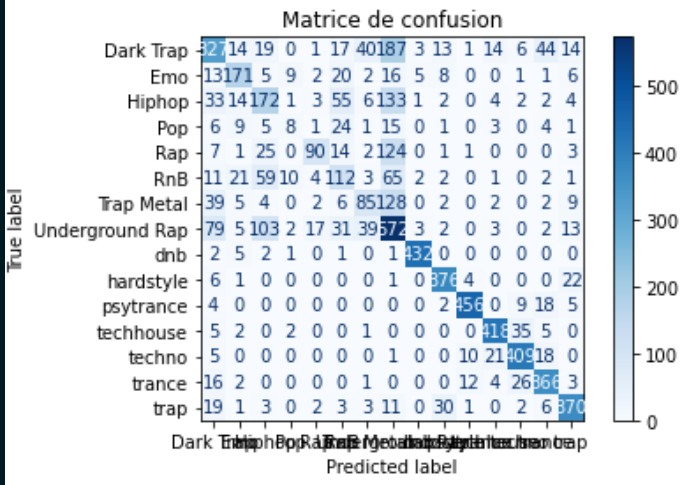
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)
```

Entraînement de différents algorithmes de classification

L'objectif est de trouver la classification qui renvoie le meilleur **F1_score**. Pour se faire, on implémente un total de 14 algorithmes, dont la majorité sont ceux proposés dans le cours *Supervised Classification* page 58.

Analyse des résultats

```
CLASSIFICATION PAR ARBRE DECISIONNEL
Précision : 56.8704695871415 %

CLASSIFICATION PAR FORET ALEATOIRE
Précision : 66.76646706586826 %

CLASSIFICATION PAR ANALYSE DISCRIMINANTE LINEAIRE
Précision : 53.38796092026473 %

CLASSIFICATION PAR ANALYSE DISCRIMINANTE QUADRATIQUE
Précision : 61.739678537661526 %

CLASSIFICATION NAIVE BAYESIENNE
Précision : 59.108099590293094 %

CLASSIFICATION PAR REGRESSION LOGISTIQUE
Précision : 58.9347620548377 %

CLASSIFICATION PAR PERCEPTRON
Précision : 40.7185628742515 %

CLASSIFICATION PAR RESEAU NEURONAL
Précision : 67.34951150330917 %

CLASSIFICATION PAR SVC
Précision : 64.90702804916482 %

CLASSIFICATION PAR BOOTSTRAP AGGREGATING
Précision : 63.55184368105894 %

CLASSIFICATION PAR EXTRA TREES
Précision : 65.74219981090451 %

CLASSIFICATION PAR ADABOOST
Précision : 41.50646076268515 %

CLASSIFICATION PAR GRADIENT BOOSTING
Précision : 68.76772770248975 %

CLASSIFICATION PAR K PLUS PROCHES VOISINS
Précision : 57.51654585565711 %
```



Scores des algorithmes utilisés

Les résultats obtenus sont assez décevants. En effet, aucun classifieur ne permet d'obtenir une bonne prédiction, c'est-à-dire avec un **F1_score** supérieur à 80%.

Pour comprendre pourquoi, on affiche la matrice de confusion pour chaque méthode. On remarque alors que la plupart des algorithmes ont énormément de difficultés pour prédire les 8 premiers genres (**Dark Trap**, **Emo**, **Hiphop**, **Pop**, **Rap**, **RnB**, **Trap Metal** et **Underground Rap**). Cela s'explique par la ressemblance de ces styles vis-à-vis des caractéristiques sonores. En effet, il est souvent difficile même pour un humain de déterminer auquel de ces styles appartient une chanson. Les autres genres sont beaucoup plus différenciés les uns des autres, ce qui facilite leur discrimination par les algorithmes.

Le classifieur le plus efficace que nous avons implanté utilise la méthode de **Gradient Boosting**, avec une précision de 68,77%.

Tentative d'amélioration

	danceability	energy	...	duration_ms	time_signature
danceability	1.000000	-0.319724	...	-0.047710	0.058500
energy	-0.319724	1.000000	...	0.293458	0.040331
key	-0.011734	0.047516	...	0.070978	0.008756
loudness	-0.213361	0.599229	...	-0.100295	0.049014
mode	0.083644	-0.032715	...	-0.033875	-0.006819
speechiness	0.180717	-0.146846	...	-0.309197	0.025848
acousticness	0.067157	-0.496061	...	-0.248978	-0.047758
instrumentalness	-0.068272	0.306910	...	0.600534	-0.011426
liveness	-0.192191	0.228010	...	0.009330	0.008837
valence	0.370821	-0.015067	...	-0.193572	0.045127
tempo	-0.169930	-0.019757	...	-0.155363	-0.021652
duration_ms	-0.047710	0.293458	...	1.000000	0.002121
time_signature	0.058500	0.040331	...	0.002121	1.000000

[13 rows x 13 columns]

Matrice de corrélation des données du fichier spotify_train

Nous avons émis l'hypothèse de réaliser une ACP sur le jeu de données afin d'obtenir de meilleurs scores de prédictions. Cependant, en affichant la matrice de corrélation, on peut remarquer que les features sont peu corrélées entre elles. Par conséquent, une ACP n'est pas pertinente dans notre cas de figure. Nous gardons alors les performances de l'algorithme précisées dans l'inventaire des scores.

Principe de l'algorithme Boosting (GradientBoostingClassifier)

L'algorithme **Boosting** permet de mettre à jour les poids afin de rééquilibrer les performances des modèles d'apprentissage. Au premier passage, les données sont classées une première fois. Si certaines des données sont mal classées, l'algorithme permet de changer la pondération et d'accorder plus de poids aux algorithmes qui ont bien classées les données au tour précédent. Ce principe est réitéré à la suite de chaque passage. En d'autres termes, les différents classifieurs sont pondérés pour qu'à chaque prédiction, ceux ayant prédit correctement les données aient un poids plus fort que ceux dont la prédiction est incorrecte au tour d'après.

Prédiction sur le jeu de données final

Maintenant que nous avons sélectionné l'algorithme le plus optimal, nous pouvons à présent l'utiliser pour prédire les genres musicaux du fichier **Spotify_test**. Les genres prédits sont rajoutés en dernière colonne du fichier .csv.

Index	s	valence	tempo	type	id	uri	track_href	analysis_url	duration_ms	time_signature	song_name	Genre prédit
0		0.766	206.402	audio_featu…	6tYL1tBaqTc…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	276813	4	All About U (ft. Nate D…	Underground Rap
1		0.351	192.07	audio_featu…	4X5AMbgW6wh…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	148093	4	Revenge	Underground Rap
2		0.517	129.976	audio_featu…	2J176Rj3ZTT…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	193542	4	Changes	trance
3		0.756	126.979	audio_featu…	08oufzbacif…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	204091	4	nan	techno
4		0.831	133.994	audio_featu…	3ZuoDMBWEnM…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	193647	5	Don't Believe Me	RnB
5		0.142	128.996	audio_featu…	1oadkytT5aV…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	421626	4	nan	techno
6		0.198	128.009	audio_featu…	2vfSCFvMbpf…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	393740	4	nan	techno
7		0.663	161.138	audio_featu…	7jLbTp3qZza…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	238053	4	Ghetto Gospel	Hiphop
8		0.586	114.97	audio_featu…	71at7DNXtys…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	151496	4	Oogabooga	Underground Rap
9		0.35	182.06	audio_featu…	2G7V7zsVDxg…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	217925	4	In My Feelings	Underground Rap
10		0.457	159.911	audio_featu…	5R2rsbwCDXO…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	166343	4	Yacht Club (feat. Juic…	Underground Rap
11		0.446	129.998	audio_featu…	72jPhFJxK6U…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	432500	4	nan	techno
12		0.375	170.152	audio_featu…	2EFWnx5OAtp…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	150569	4	White Parents Are Gonna H…	Trap Metal
13		0.334	197.736	audio_featu…	7aRNccA1RsV…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	246147	4	Tell Me (feat. Chri…	Hiphop
14		0.0336	128.014	audio_featu…	7fr4aNfPCFY…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	459380	4	nan	techno
15		0.198	134.027	audio_featu…	2GAYbGqYbkJ…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	204301	4	nan	techno
16		0.515	170.046	audio_featu…	1Au9zBd86mt…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	165982	4	BloodyBoysW…	Underground Rap
17		0.168	140.029	audio_featu…	28FJ6nAd3si…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	296677	4	nan	techno
18		0.422	150.04	audio_featu…	09wXMeucpfd…	spotify:tra…	https:// api.spotify…	https:// api.spotify…	198429	4	nan	techno

*Dataframe **Spotify_test** avec les genres prédits*

Exercice 2

Principe de l'exercice

Le but de l'exercice est d'entrainer notre algorithme à prédire la popularité des musiques disponibles dans la base de données de Spotify. Etant donné le caractère continu des données de popularité, nous optons pour une méthode de régression. La performance des algorithmes testés sera quantifiée selon le **RMSE** et le $\mathbf{R^2}$.

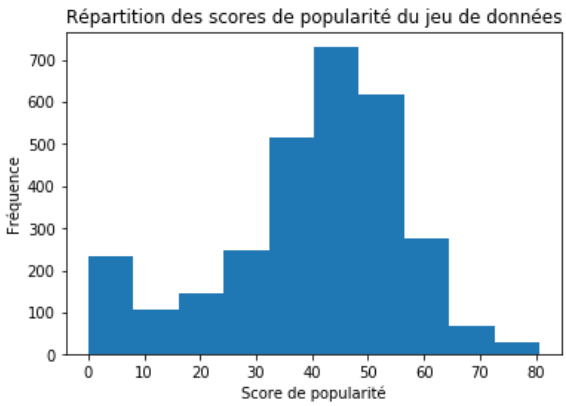
Analyse du jeu de données d'entraînement

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              2973 non-null   int64
 1   genres            2973 non-null   object
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64
dtypes: float64(11), int64(2), object(1)
```

Le jeu de données d'entraînement est un DataFrame composé d'environ 3 000 musiques caractérisées selon 14 attributs. Le 13e (**popularity**) est celui que nous cherchons à prédire. Parmi les 13 autres, l'attribut **genres** contient des chaines de caractères. Chaque musique possède un genre spécifique différent des autres, ce qui rend cette colonne inutile pour la classification. En effet, le genre n'est dans cette situation pas une donnée discriminante, mais fait office de "description". Pour remédier à ce problème, on peut tenter de regrouper les genres similaires entre eux. Au vu de leur dénomination, ce travail est presque impossible, et nous avons donc préféré nous séparer de cet attribut pour la suite de l'exercice.

Après traitement, le jeu de données "efficace" contient alors 12 catégories et la popularité associée, qui sont toutes de type *float* ou *int*. Les données n'ont pas besoin d'être nettoyées, ce qui facilite le traitement.

Répartition des données de popularité



L'étude de l'histogramme des scores de popularité montre une répartition approximativement gaussienne des valeurs, avec un maximum de fréquence pour des scores d'environ 45.

Traitement préliminaire des données

On définit **x** le tableau de vecteurs composé des 12 attributs, et **y** le celui de la popularité.

```
y = df.iloc[:, 12]
y = pd.Series(y).to_numpy()
y = y.reshape(2973,-1)

x = df.drop(['popularity', 'genres'], axis=1)
```

Pour augmenter l'efficacité des algorithmes de classification, on choisit de réduire et centrer les données de **x** en utilisant, comme pour le premier exercice, la fonction **StandardScaler()** du module preprocessing de la bibliothèque Scikit_learn.

```
sc = StandardScaler()
x = sc.fit_transform(x)
```

Tous les tests ont été effectués sur le jeu d'entraînement avec le même mélange des données : la proportion pour l'apprentissage a été fixée à 80%, et 20% pour le test. De plus, le mélange a été réalisé selon le même **Random_State**.

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)
```

Entraînement des données

L'objectif est de prédire la popularité d'une chanson et donc d'implémenter un algorithme de régression. Pour se faire, nous avons essayé plusieurs algorithmes de régression comme **Ridge**, **SVM** ou encore le Perceptron multicouches **MLPRegressor**. Nous allons ici confronter ces algorithmes et garder le plus pertinent pour la prédiction de la popularité des chansons.

Description et explication du RMSE et du R2

Le **RMSE** (l'erreur quadratique moyenne) mesure la différence entre les valeurs prédites par l'algorithme de régression et les valeurs observées du label qu'on souhaite prédire. Compris entre 0 et 1, plus le RMSE est faible et proche de 0, plus l'algorithme s'ajuste parfaitement aux données, et effectue la prédiction avec une bonne précision.

Le $\mathbf{R^2}$ (coefficient de détermination) mesure l'adéquation entre le modèle et les données observées. Il s'agit du rapport de la somme des résidus au carré, où les résidus sont les écarts entre les observations y_i et les valeurs prédites, sur la somme des écarts entre les y_i et leur moyenne au carré. S'il vaut 1, la droite de régression détermine 100% de la distribution des points. Si au contraire il vaut 0 ou s'il est négatif, le modèle de régression considéré ne correspond pas aux données.

Analyse des résultats

```
REGRESSION PAR RESEAU NEURONAL (MULTILAYER PERCEPTRON)
RMSE : 11.659298945426286
R2 : 0.5182773726861305

REGRESSION VECTORIELLE (SVM)
RMSE : 12.886812644998775
R2 : 0.4115044273955586

REGRESSION NAIVE DE BAYES
RMSE : 14.5582940715006
R2 : 0.24894257117806207

REGRESSION RIDGE
RMSE : 14.544781054385295
R2 : 0.25033618799366686
```

Scores de tous les algorithmes utilisés

On constate que parmi les algorithmes précédents, le **MultiLayer Perceptron** est mieux adapté à la régression car donne les meilleurs scores. En effet, on obtient un **RMSE** de 11.66, et un **R^2** de 0,52.

Tentative d'amélioration

	mode	acousticness	...	popularity	key
mode	1.000000	0.077347	...	-0.031231	-0.060109
acousticness	0.077347	1.000000	...	-0.458698	-0.078815
danceability	-0.071240	-0.324991	...	0.217992	0.029551
duration_ms	-0.053197	-0.014075	...	-0.071019	-0.029309
energy	-0.080352	-0.868274	...	0.337795	0.097165
instrumentalness	-0.012662	0.273739	...	-0.265449	-0.069317
liveness	-0.002650	-0.076570	...	-0.094178	0.007308
loudness	-0.063706	-0.737276	...	0.344361	0.082991
speechiness	-0.059127	-0.086139	...	-0.045217	0.022326
tempo	-0.042931	-0.397713	...	0.146717	0.081608
valence	-0.010258	-0.178210	...	0.023072	0.063637
popularity	-0.031231	-0.458698	...	1.000000	0.008577
key	-0.060109	-0.078815	...	0.008577	1.000000

[13 rows x 13 columns]

Matrice de corrélation des données du fichier Spotify_exo2

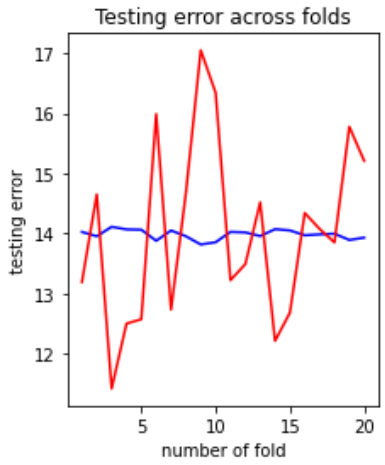
Pour les mêmes raisons que dans l'exercice 1, la matrice de corrélation nous montre qu'une ACP n'est pas pertinente.

Confrontation des performances

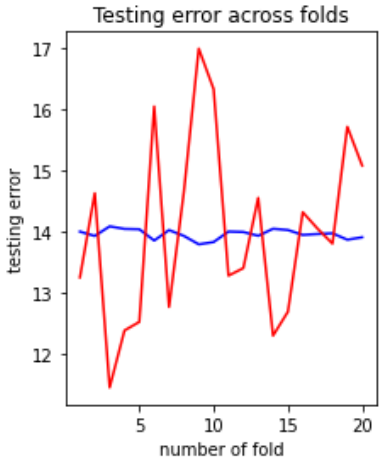
A ce stade, on voudrait s'assurer que ces bons scores ne cacheraient pas un over-fitting durant l'entrainement du jeu de données. Pour cela, on étudie différents graphes.

- Pour un nombre d'itérations données, on trace l'évolution du RMSE en fonction du nombre de paquets qui divisent le jeu de données au sein d'une validation croisée, en utilisant la fonction **verif_over()**.

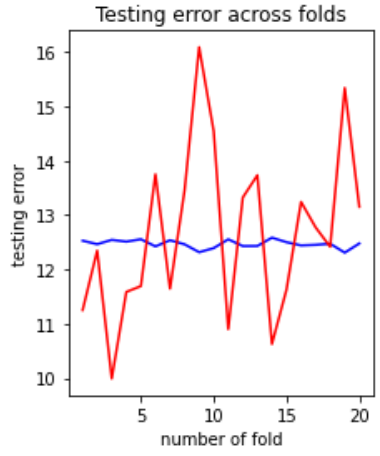
RIDGE



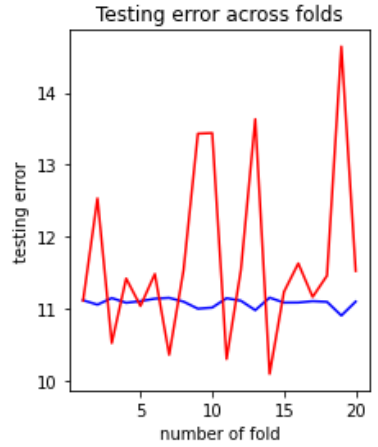
BAYESIAN RIDGE



SVM



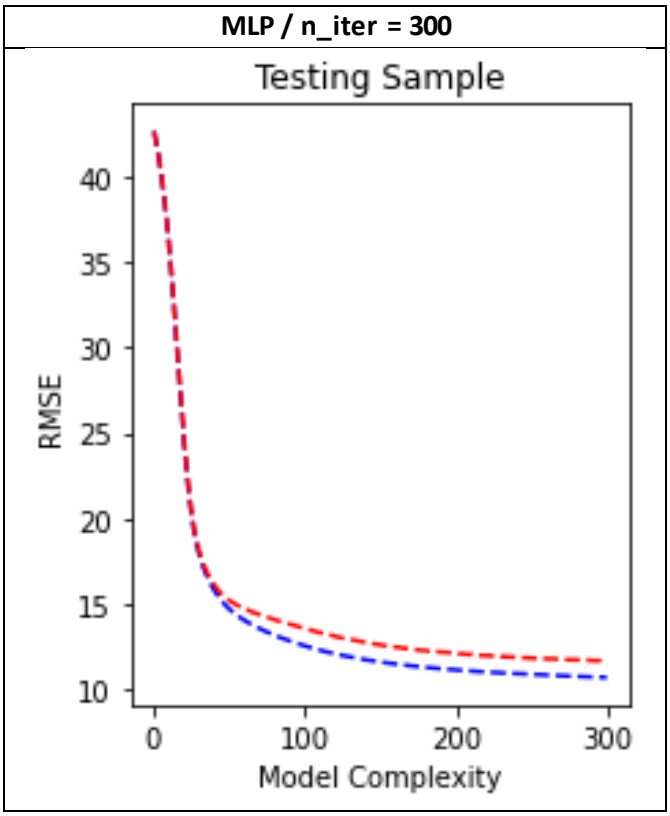
MLP



*En bleu est représentée l'évolution du **RMSE** pour le jeu d'apprentissage, et en rouge pour le jeu de test.*

On remarque alors que l'intégralité des algorithmes implémentés réalisent de l'over-fitting, néanmoins il semblerait que le **MultiLayer Perceptron** soit celui qui en effectue le moins.

- On se propose également d'analyser l'évolution des prédictions en fonction de la complexité, en utilisant la fonction **complexity()**.



*En bleu est représentée l'évolution du **RMSE** pour le jeu d'apprentissage, et en rouge pour le jeu de test.*

On observe bien l'apprentissage de l'algorithme, car le RMSE diminue fortement pour les premières valeurs d'itérations.
On constate que les courbes bleues et rouges sont relativement bien superposées, ce qui signifie que le modèle ne subit que peu d'over-fitting pour ces valeurs d'itérations.

Prédiction sur le jeu de données final

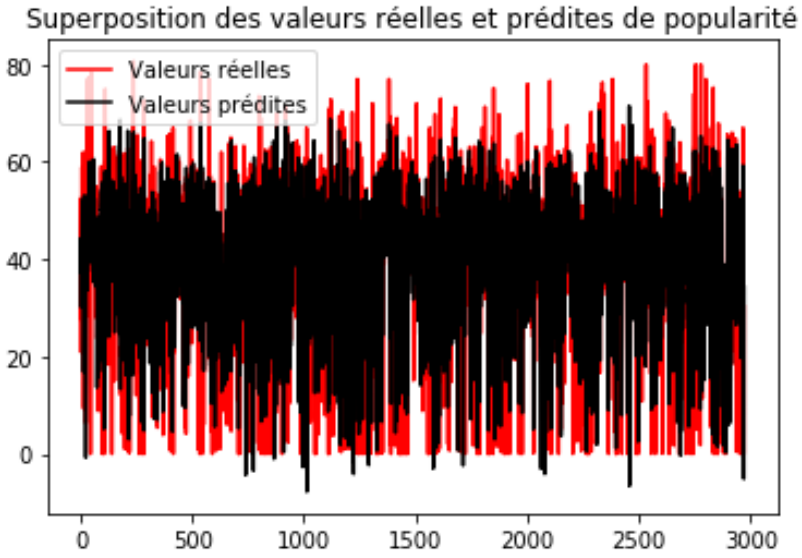
Il semblerait alors que **MultiLayer Perceptron** permet de prédire au mieux les scores de popularité des chansons. C'est donc cet algorithme qui sera choisi pour effectuer la prédiction finale des popularités.

Son implémentation donne alors les caractéristiques suivantes pour la prédiction :

- Un **RMSE** de 11,66
- Un $\mathbf{R^2}$ de 0.52

Comparaison des prédictions aux valeurs réelles

On se propose maintenant d'évaluer la performance de la prédiction de manière graphique, en traçant l'écart de la popularité prédite avec la valeur réelle, grâce à la fonction **trace_comparaison()**. L'écart observé se traduit évidemment par un **RMSE** non nul. Néanmoins, les résultats semblent relativement "proches" de la réalité.



Conclusion

L'enjeu du travail de DATA Scientist réside dans sa capacité à sélectionner le bon modèle de classification, c'est-à-dire le plus adapté au jeu de données qu'il doit analyser. Ce projet nous a permis de se confronter à cette problématique. Nous avons du sélectionner un modèle parmi ceux existant, puis analyser ses performances dans la prédiction des jeux de données issus de l'application Spotify®.