Rapport Projet IAS

Chargé de TP référent : Mr Sanchez.

Étudiants : MAQUET Baptiste || VINCENT Maxime || JEROMIN Tristan
MANOUNOU Reine || AIT MOUFFOK Letitia.

Introduction :

Dans le cadre de l'UE Introduction à l'apprentissage statistique, nous devions nous regrouper par groupes de 4-6 maximum et choisir un sujet libre qui nous passionne. Ce projet nous a permis d'appréhender les enjeux du processus de développement d'un projet de machine learning, les bases en apprentissage statistique et de bien maîtriser les fondamentaux. Nous avons été confrontés à des problèmes d'analyse de données, modélisation, d'apprentissage automatique et de prédiction. Le tout à l'aide de bibliothèques python spécifiques au machine learning. Nous allons détailler dans ce rapport comment nous avons fait face à ces défis et quelles méthodes nous avons employées.

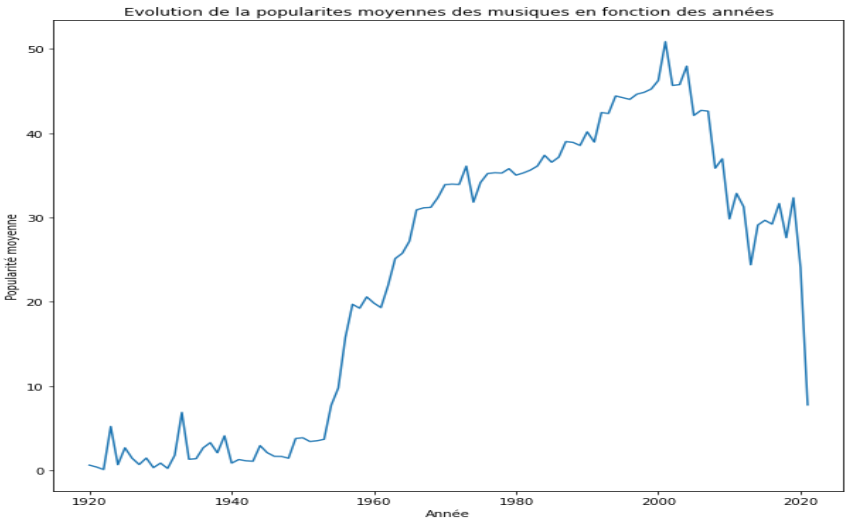
Thème :

Dans le cadre du projet IAS, nous avons choisi de créer un modèle qui prédit la popularité d'une musique Spotify. Nous avons récupéré nos données sous format csv sur Kaggle depuis https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks.

Exploration des données :

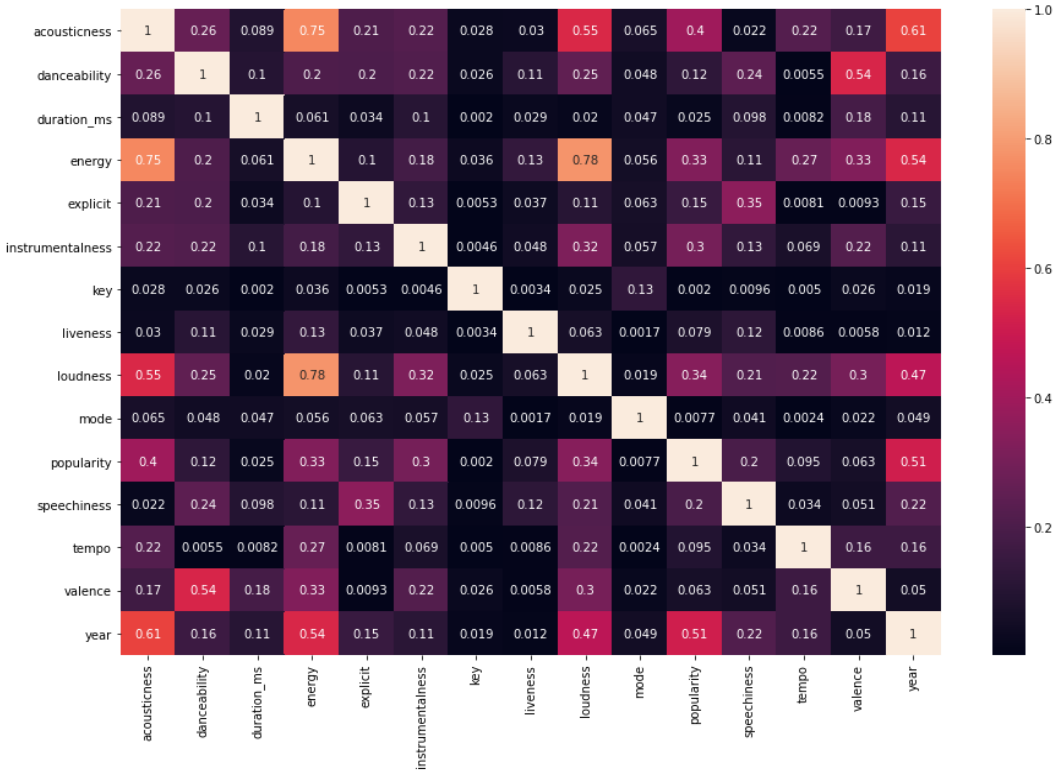
Le dataset comporte 174389 individus (lignes) et 19 attributs/features (colonnes) sans aucune donnée manquante. Chaque individu représente une musique, avec sa liste d'attributs : acousticness, artists, danceability, duration_ms, energy, explicit, id, instrumentalness, key, liveness, loudness, mode, name, popularity, release_date, speechiness, tempo, valence, year.
Les noms des attributs sont bien significatifs. Pour plus de détails, vous pouvez consulter le lien Kaggle.
Nous voudrions pouvoir prédire l'attribut "**popularity**" qui prend ses valeurs entre 0 et 100 (valeurs entières) en partant des autres attributs d'une musique donnée.
Notez que l'attribut popularity est décrit comme : *'The popularity of the song lately, default country = US'*.

Pour mieux comprendre la popularité des musiques de notre jeu de données, nous avons tracé des graphiques qui mettent en avant le lien entre la popularité *"popularity"* et l'année de sortie de la musique *"year"* :



On remarque grâce à ces deux visualisations que l'année 2001 est celle avec le plus de musiques populaires en moyenne.
L'intérêt d'utiliser la moyenne est d'éviter que les années où il y a eu une sur-production de musiques soient bien classées seulement en raison de la grande quantité produite.
On peut expliquer la popularité des musiques de 2001 par tous les titres populaires et les grandes collaborations qui ont eu lieu plus précisément dans la musique *pop*. (article)

Nous avons ensuite affiché la heatmap de la matrice de corrélation (en valeur absolue pour plus de visibilité) afin de voir les colonnes qui circulent des informations similaires (exemple : |Corr(*energy*, *loudness*)|=0.78, |Corr(*energy*, *acousticness*)|=0.75).
Cependant, nous n'avons pas fusionné à la main les features mais nous avons décidé d'utiliser la PCA (Analyse en Composantes Principales) pour effectuer une réduction de dimension automatiquement.



Preprocessing :

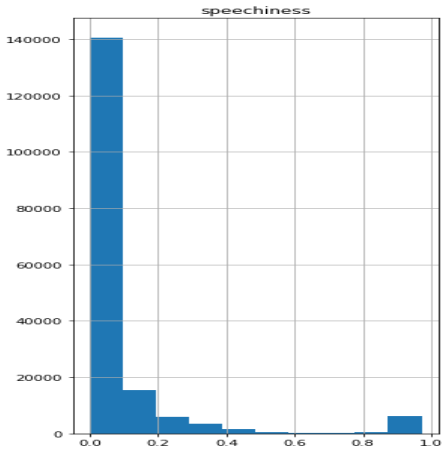
Dans cette partie, nous avons procédé au nettoyage des données en utilisant différentes approches :

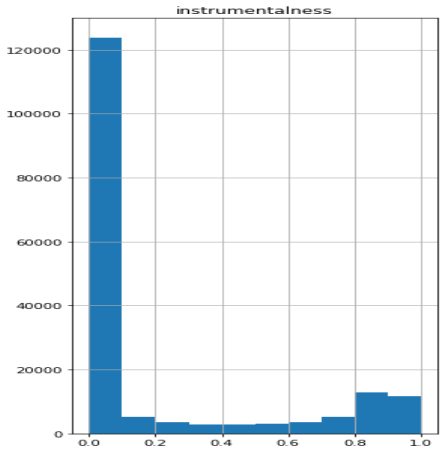
Suppression d'attributs :

On a supprimé la colonne *"id"* car c'est un identifiant unique qui n'est présent que pour les besoins de la base de données en tant que clé primaire.
La colonne *"name"* n'étant pas une donnée catégorielle, nous devions effectuer une méthode de TAL (Traitement Automatique des Langues), mais nous l'avons simplement supprimée pour ne pas rendre les choses plus complexes.
La feature *"artists"* pouvait être transformée en la moyenne de popularité de l'artiste, mais à cette étape nous avions décidé de la supprimer.
Enfin, nous avons remarqué que la colonne *"release_date"* était juste une précision de la colonne *"year"* et qui en plus de ça ne suit pas un format standard, nous l'avons alors supprimée aussi.

Transformation d'attributs :

Comme le montrent les figures ci-dessous, les features *"speechiness"* et *"instrumentaliness"* sont soit proches de 0 ou de 1, on les a donc converties en des valeurs binaires en prenant 0.5 comme seuil (threshold).



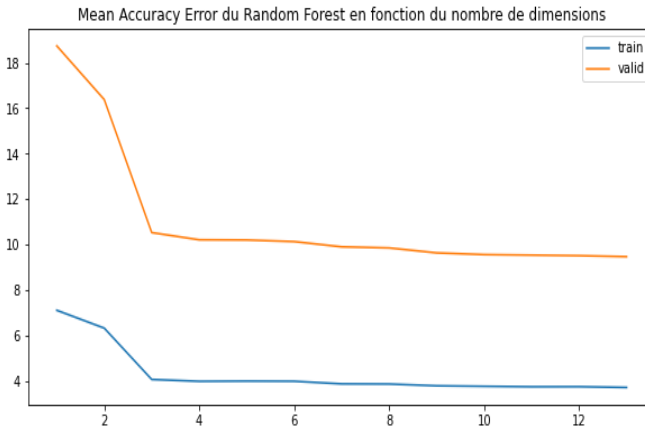


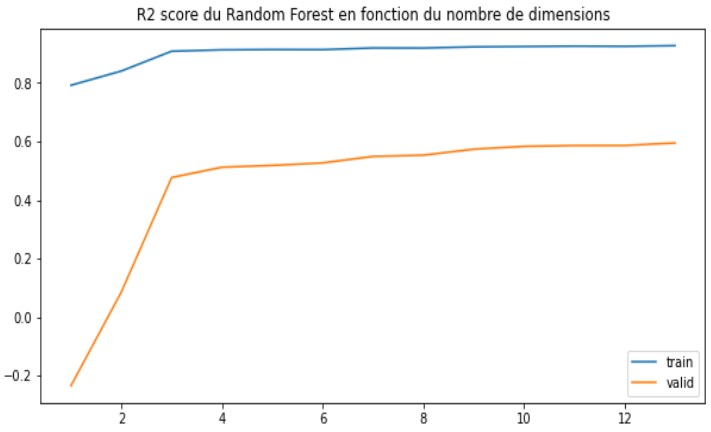
Suppression de doublons :

Nous avons supprimé les individus (lignes) dupliqués avec la fonction pandas *drop_duplicates()*

Réduction de dimensionnalité par PCA :

Dans cette partie, nous avons séparé notre dataset en 3 parties : jeu d'apprentissage, jeu de validation et jeu de test.
Soit "n" le nombre de colonnes restantes dans notre jeu de données.
Nous avons ensuite essayé une décomposition en 1 jusqu'à n composantes principales, et grâce à un régresseur RandomForest, nous avons choisi le nombre de composantes principales qui permettrait à ce régresseur d'avoir de bonnes performances sur le jeu d'entraînement et sur le jeu de validation en *"Mean Accuracy Error"* et en *"R2 Score"*.
Les résultats ont été les suivants :

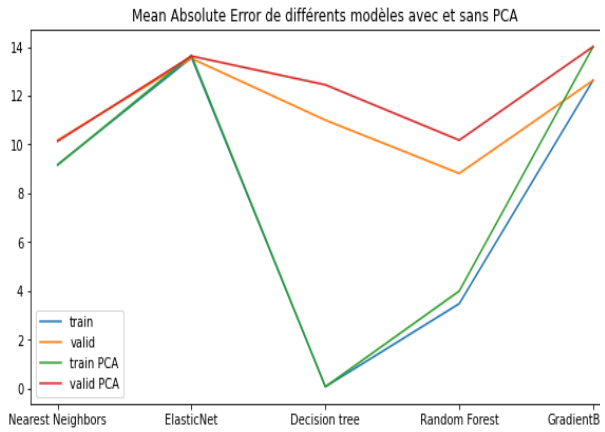


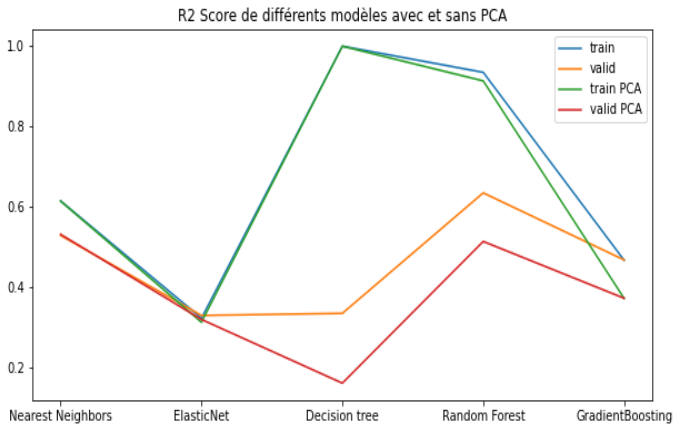


La PCA nous indique que seulement 3-4 dimensions sont nécessaires pour obtenir le meilleur score. Au-delà le gain est anecdotique. On a donc largement réduit le nombre de dimensions. Cela peut s'expliquer par le fait que des groupes de features se sont fusionnés ensemble, ce qu'on ne pouvait pas voir avec la matrice de corrélation. Notre choix s'est porté sur 4 composantes.

Modélisation et sélection du meilleur modèle :

Dans cette partie, nous avons défini une liste de modèles de régression importés de Scikit Learn comportant : un K Neighbors Regressor avec k = 10 (nombre de voisins les plus proches à regarder), un Elastic Net, un Decision Tree Regressor, un Random Forest Regressor avec 10 estimateurs et enfin un Gradient Boosting Regressor avec 10 estimateurs aussi.
Le but est de parcourir tous ces modèles et de voir leurs performances (en *Mean Absolute Error* et en *R2 score*) sur les jeux de données d'apprentissage et de validation avant la transformation avec PCA et après la transformation (4 jeux de données en tout) après les avoir entraînés sur les jeux d'apprentissage (avec et sans PCA), puis de choisir le modèle qui s'adapte le mieux aux données d'entraînement et se généralise le mieux aux données de validation.
Nous avons sauvegardé les scores dans des listes pour pouvoir réaliser les visualisations suivantes et comparer les résultats :

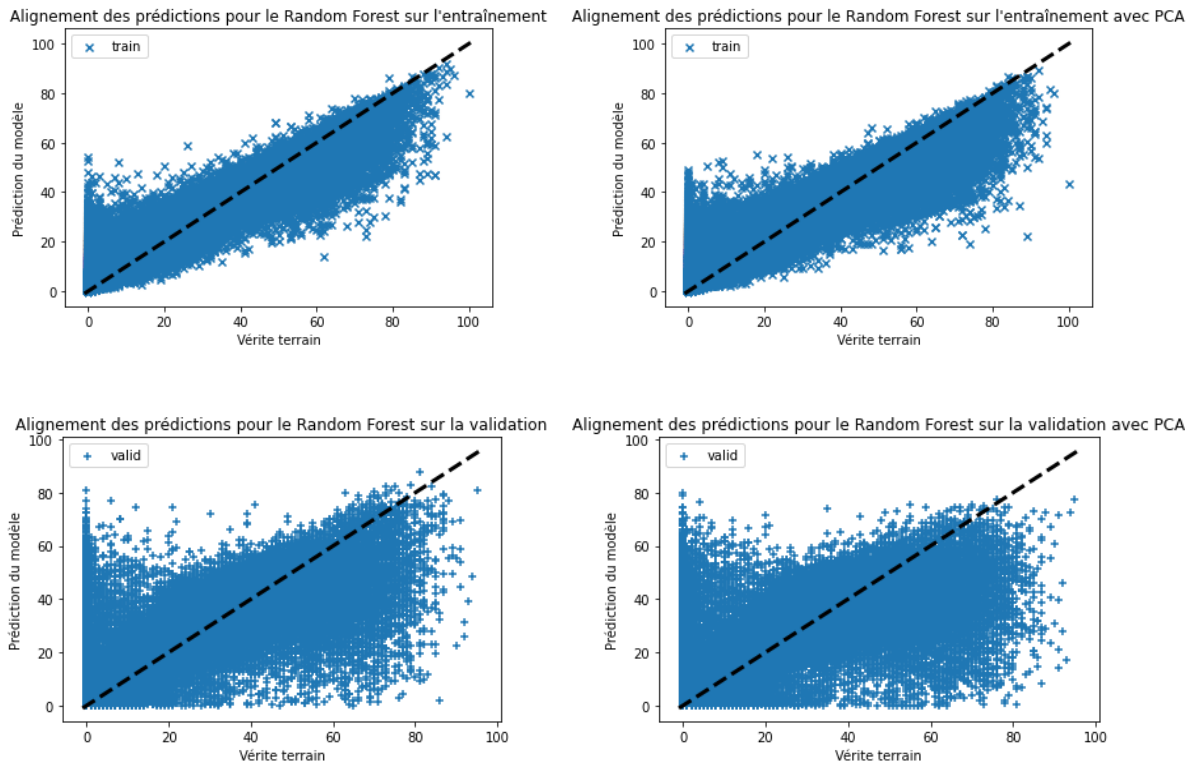




Discussion des résultats :

Les modèles ayant le meilleur score sur les jeux d'entraînement sont le Decision Tree et le Random Forest. Le Decision Tree dépasse de quelques points de score du Random Forest sur l'entraînement. En revanche, le Decision Tree généralise très mal car les résultats sur la validation sont bien en dessous de ceux obtenus avec le Random Forest. Les autres modèles ont de mauvais scores et certains sont en underfitting.
Finalement, nous avons choisi le Random Forest qui semble donner les meilleurs résultats sur notre jeu de données.
Autre remarque : on a obtenu des résultats similaires avec et sans PCA pour l'ensemble d'apprentissage. Cependant, sur l'ensemble de validation avec la PCA on généralise moins bien qu'avec toutes les features. La différence est d'environ 10%. Nous avons donc par la suite utilisé le jeu de données transformé par PCA en espérant compenser cette différence de score à l'aide de la recherche des hyper-paramètres du Random Forest.

Nous avons utilisé une autre visualisation pour voir la dispersion des prédictions du Random Forest sur nos jeux d'entraînement et de validation. Ci-dessous les résultats :

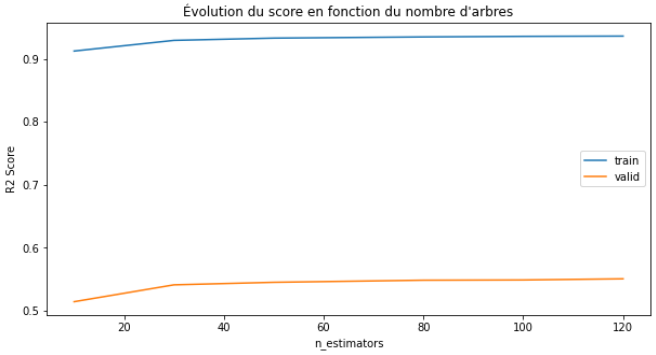


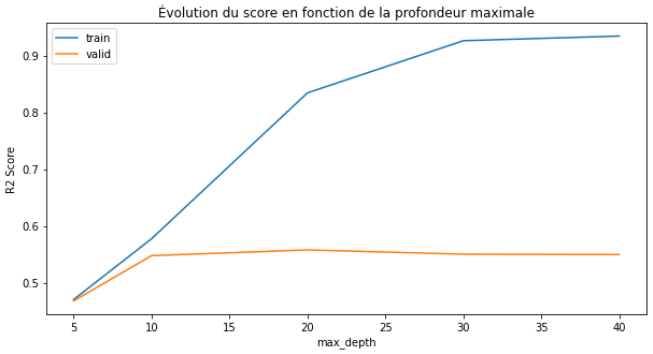
Ces nuages de points confirment les résultats obtenus précédemment.
On remarque que pour les ensembles de validation, les points sont très épars et s'éloignent beaucoup de la droite identité.

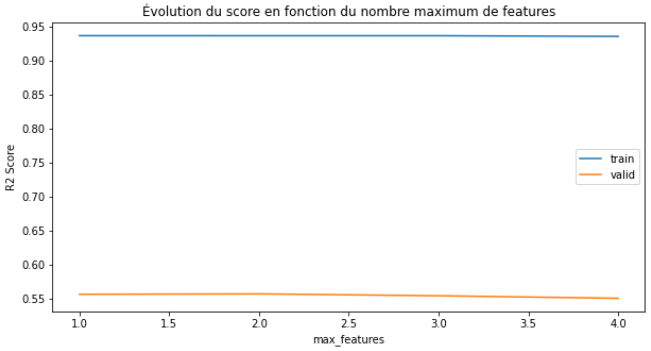
Choix des hyperparamètres :

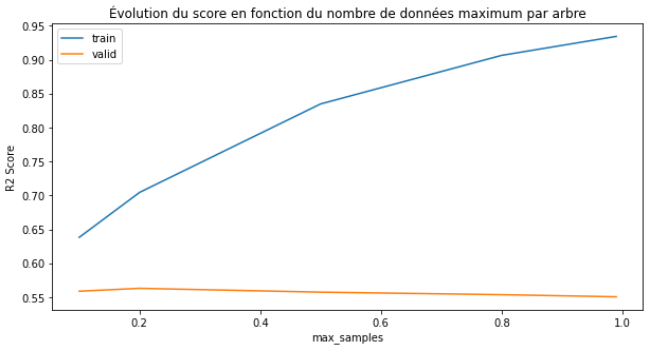
Dans cette partie, nous avons essayé de trouver les meilleurs hyperparamètres pour notre Random Forest Regressor. Les hyperparamètres que nous avons tunés manuellement sont *n_estimators*, *max_depth*, *max_features*, *max_samples* et *min_samples_split* en essayant

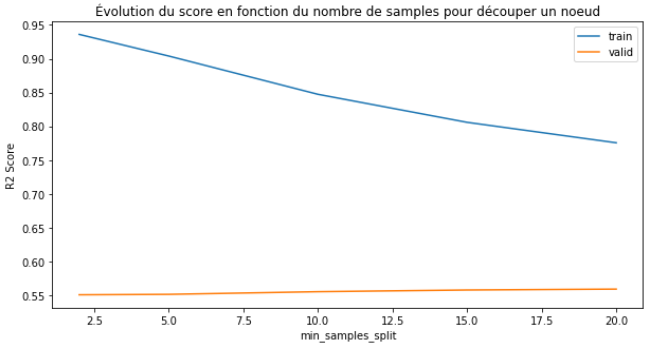
une plage de valeurs pour chacun d'eux puis prendre la valeur qui nous offre le meilleur score sur le train et valid set avec PCA. Nous avons aussi choisi d'utiliser le bootstrap ou pas en utilisant la fonction *"GridSearchCV"*.
Pour ces 6 hyperparamètres nous avons obtenu les résultats suivants :

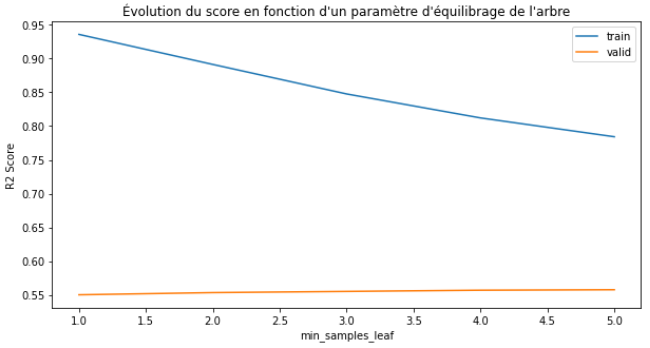












Après tous nos essais, nous avons trouvé que notre modèle était déjà bien paramétré par défaut pour éviter l'overfitting.

Evaluation :

En essayant notre modèle sur le jeu de test, nous avons obtenu comme *r2 score* : 0.546 et comme *mean absolute error* : 9.89. Ce ne sont pas des résultats dont nous pouvons être fiers.

Pipeline final :

Dans le pipeline final, nous avons décidé de revenir sur la colonne *"artists"* surlignée précédemment et au lieu de la supprimer on a remplacé les artistes par des valeurs numériques qui représentent leur moyenne de popularité (cela prenait ~5 min) et en gardant exactement les mêmes étapes pour le reste du preprocessing.

En réévaluant notre modèle sur le jeu de test, nous avons obtenu une hausse époustouflante en performance : ***r2 score*** **: 0.85** ***mean absolute error*** **: 5.38**

Cette hausse en performance aura été possible grâce à ce bout de code :

```
# On remplace les artistes par des valeurs numériques
# représentant leur moyenne de popularité
artistByPopularity = data.groupby('artists')['popularity'].mean()
data = data.replace({'artists': artistByPopularity})
```

Conclusion :

Ce projet nous a permis d'appliquer nos connaissances théoriques acquises durant les cours de l'UE Introduction Apprentissage Statistiques et les mettre en pratique, nous avons ainsi pu approfondir nos connaissances en Machine Learning et apprendre également à bien en maîtriser les fondamentaux.