****

Au fil des années, une multitude de chansons se sont placées en tête des classements musicaux les plus célèbres. On constate que chaque époque a été marquée par un courant musical.

Pour quelles raisons une chanson plaît-elle à un public précis ? Pourquoi certaines chansons ne rencontreront-elles jamais aucun succès avec ce même public  ?

A notre époque la musique est principalement consommée sur en ligne et l’industrie musical investit de plus en plus pour comprendre les données des utilisateurs en ligne. Cette grande quantité de données permet d’étudier l’impact d’un artiste ou d’une chanson et proposer des titres musicaux à tous les types d’utilisateurs.

Il est désormais possible d’extraire la probabilité qu’une chanson plaise à un utilisateur grâce à **SOUNDSGOOD**.

Avec cette application, nous pouvons désormais repérer les titres musicaux et les artistes de demain.

En lisant cet article, vous découvrirez comment **SOUNDSGOOD** a été conçu.

Étape 1 : le choix du DataSet et Datacleaning

**Dataset**

Nous avons sélectionné Top 50 Spotify Song – 2019 en guise de dataset. Il contient des données sur les 2700 titres les plus populaires de l’année 2019 sur Spotify.

On retrouve dans ce dataset 15 colonnes différentes : *ID, Title, Artiste, Top Genre, year, bpm, nrgy, dnce, dB, live, val, dur, acous, speechiness, pop.*

L’*ID* est l’identifiant de la chanson dans le dataset. Les colonnes *Title,* *Artist*, *Top genre* et *Year* contiennent respectivement les titres des chansons, les noms des artistes, le genre de la chanson et l’année de sortie dans le Billboard.

*Tempo* indique le rythme de la chanson.

*Energy* indique les chansons qui sont perçues comme rapides, fortes et bruyantes. Par exemple une chanson rock aura une note plus élévée sur l’échelle de l’énergie qu’une chanson classique.

*Danceability* correspond à l’aptitude à danser sur le morceau de musique, plus la note est élevée, plus il est simple de danser sur cette chanson.

*Loudness* correspond à l’intensité sonore de la chanson.

*Liveness* indique la présence d’un public lors de l’enregistrement. Plus la note de vivacité est élevée, plus il y a des chances que la chanson ait été jouée en direct.

*Instrumentalness* est la mesure qui définit l'utilisation d'instruments dans la musique.

*valence* indique la positivité ressentie lors de l’écoute de la chanson.

*Duration\_ms* correspond à la durée de la chanson.

*Acousticness* indique si le son est acoustique.

*Speechiness* détecte la présence de mots parlés dans une chanson : plus il est élévé et plus il y a de mots parlés.

*Key* la clé musicale globale estimée de la chanson.

*Mode* indique la modalité d’un morceau (majeure ou mineure) d’un morceau.

*Time\_signature* est une estimation de la signature rythmique globale (nombre de battements) d’une piste.

*Target* indique si le user apprécie ou non la chanson.

Quel ordre pour le datacleaning ? Partie sur les features à préciser. Lignes inutiles à supprimer. PLUSIEURS familles de données créer : Supprimer celles qui sont inutiles

**Datacleaning**

Beaucoup de features ont des similitudes. Par exemple, *instrumentalness* est la mesure qui definit l'utilisation d'instrument dans la musique. Tandis que la *speechiness* mesure l'utilisation de la voix comme nous l’avons évoqué précédemment.

Les deux features mesurent donc la musicalité d’une chanson.

Ainsi nous avons créé de catégories de features :

* Class Labels : target
* Interval : acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, valence
* Time : duration\_ms
* Numerical : tempo
* Ordinal : key, time\_signature
* Binary : mode
* String : song\_title & artist

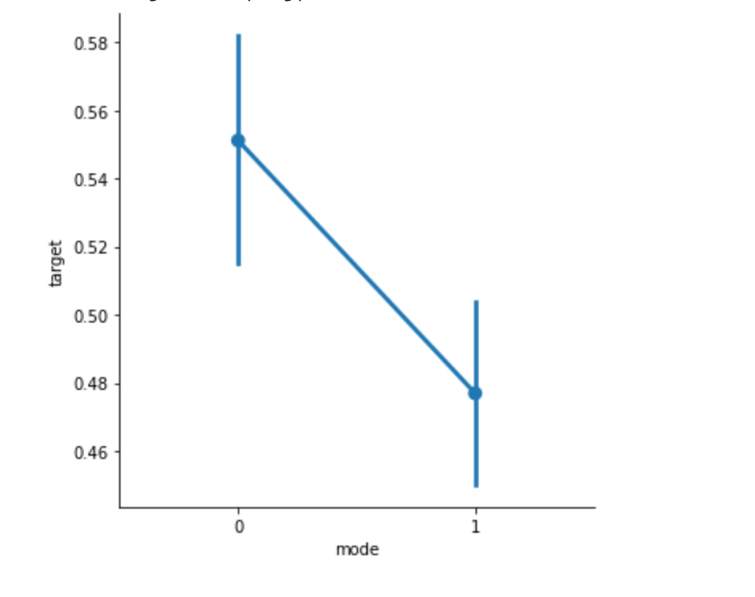
A l’aide de la méthode binning, nous avons créé de nouvelles features à partir des familles que nous avons créées précédemment.

* instru\_ind : L'instrumentale d'une musique et ses paroles
* beat\_ind : L'énergie et le beat d'une musique
* melody\_ind : Toute la mélodie d'une musique
* env\_ind : L'ambiance d'une musique
* valence

Nous avons donc décidé de nettoyer notre dataset afin de conserver uniquement les colonnes nécessaires pour travailler sur la prédiction de la popularité d’une chanson auprès d’un user.

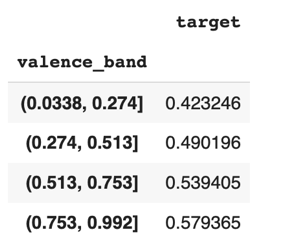
Pour ce fait, nous avons remplacés les caractères string des colonnes song\_title, artiste sont remplacés par la valeur nulle.

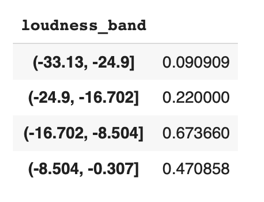
Nous avons par la suite procédé à une analyse des features de manière graphique *(by visualisation).* En effet, nous avons étudié la courbe représentative de target en fonction du mode.



On constate que plus le *mode* (qui indique la modalité d’un morceau (majeure ou mineure) d’un morceau) augmente et plus la *target* (qui indique si le user apprécie ou non la chanson) diminue.

Par la suite, il était important d’étudier la valeur de nos target en fonction des valeurs de nos features. Nous avons créé des intervalles de données de valeurs pour chaque feature avec la valeur de target associée.

Une image contenant capture d’écran

Description générée automatiquementUne image contenant capture d’écran

Description générée automatiquementUne image contenant capture d’écran

Description générée automatiquementUne image contenant capture d’écran

Description générée automatiquementUne image contenant capture d’écran

Description générée automatiquementUne image contenant capture d’écran

Description générée automatiquement

Une image contenant capture d’écran

Description générée automatiquement

Nous avons ainsi une vision des indicateurs qui permettent au score de la target d’augmenter et ceux qui le font diminuer.

Nous avons donc décidé de nettoyer notre dataset afin de conserver uniquement les colonnes nécessaires pour travailler sur la prédiction de la popularité d’une chanson auprès d’un user. Nous supprimons les features les moins performantes : *'instru\_band', 'speech\_band', 'dance\_band', 'energy\_band', 'acoustic\_band', 'liveness\_band', 'loudness\_band', 'tempo\_band', ,'valence\_band'.*

.

Il faut également savoir que notre modèle n’accepte que la data type binaire/nombres entiers or la plupart de nos données sont des nombres décimaux.

Pour ce fait, nous avons remplacés les caractères string des colonnes song\_title, artiste sont remplacés par la valeur nulle.

On crée de nouveaux regroupements de features à partir des familles déjà créees auparavant avant de les regrouper dans les intervalles de valeur.

* instru\_ind : instrumentalness & speechiness
* beat\_ind : danceability & energy
* melody\_ind : tempo, mode, key, time\_signature
* env\_ind : acousticness, liveness, loudness
* Valence

Par la suite nous créons des valeurs dummies pour toutes les colonnes sauf la colonne *target*.

**Étape 2 : Modele, prediction & evaluation**

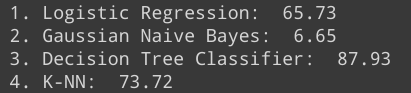
Nous allons baser notre travail sur les modèles suivants :

* Logistique Regression qui est utilisé pour modéliser la probabilité de l'existence d'une certaine classe ou d'un certain événement tel que réussite/échec, gagnant/perdant, vivant/mort ou sain/malade. Dans notre étude, il nous permettra de modéliser la probabilité qu’une chanson plaise à un utilisateur.
* Gaussian Naive Bayes est un type de classification bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses.
* Decision Tree Classifier utilise un arbre de décision (comme modèle prédictif) pour passer des observations sur un élément (représenté dans les branches) aux conclusions sur la valeur cible de l'élément (représentée dans les feuilles).
* K-NN : C’est un algorithme qui peut servir autant pour la classification que pour la régression. Il est surnommé « nearest neighbors » (plus proches voisins, en français) car le principe de ce modèle consiste en effet à choisir les **k** données les plus proches du point étudié afin d’en prédire sa valeur.

On obtient pour le **training** les résultats suivants

On a donne un maximum de données à notre modèle. On forme notre modèle signifie simplement apprendre (déterminer) de bonnes valeurs pour tous les poids et les biais à partir d'exemples étiquetés. Dans l'apprentissage supervisé, un algorithme d'apprentissage automatique construit un modèle en examinant de nombreux exemples et en essayant de trouver un modèle qui minimise les pertes ; ce processus est appelé minimisation empirique des risques.

Quelles sont les données justes ?

[](https://cdn.discordapp.com/attachments/694920085122580482/702645796155162645/unknown.png)

**Testing**

Pour la partie testing nous obtenons les résultats suivants, nous testons notre modèle avec de nouvelles valeurs :

1. Logistic Regression: 61.39

3. Decision Tree: 60.23

4. K-NN: 61.22`

Une image contenant bouteille, noir, signe, blanc

Description générée automatiquement

**K-Kross validation :** Cette méthode va nous permettre d’estimer la compétence de notre machine à apprendre.

1. Logistic Regression: 56.79

3. Decision Tree Classifier: 56.97

4. K-NN: 57.26

[Une image contenant bouteille, noir, blanc, signe

Description générée automatiquement](https://cdn.discordapp.com/attachments/694920085122580482/702645955240919100/unknown.png)

**Création de l’interface SoundsGood**

Nous avons créé une interface ludique et simple d’utilisation à laquelle nous avons implémenté notre modèle de prédiction. Ainsi, nous avons réalisé une application en React que nous avons déployée sur Heroku.

Une image contenant intérieur, table, assis, fermer

Description générée automatiquement

Grâce à l’application SoundsGoods, vous pouvez en quelques clics connaître le potentiel d’une chanson à être appréciée par un utilisateur.

Tout d’abord il faut d’entrer le titre de la chanson de notre choix dans le champ dédié puis la plausibilité que la chanson plaise à notre utilisateur sera renvoyée en fonction des différents modèles de prédiction avec lesquels nous avons travaillé.

Une image contenant moniteur, intérieur, assis, table

Description générée automatiquement

Rejoignez-nous vite sur SoundsGood !