**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

### Marina Rocha Parente

**Nuvem de Músicas -**

**Criando playlists agrupadas utilizando o conceito de clusterização**

Brasília

**Nuvem de Músicas -**

**Criando playlists agrupadas utilizando o conceito de clusterização**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Sumário

[1. Introdução 4](#_Toc69649670)

[1.1. Contextualização 4](#_Toc69649671)

[1.2. O problema proposto 5](#_Toc69649672)

[1.3 Tecnologias utilizadas 6](#_Toc69649673)

[2. Coleta de Dados 7](#_Toc69649674)

[3. Tratamento de Dados 15](#_Toc69649675)

[4. Análise e Exploração dos Dados 17](#_Toc69649676)

[5. Criação de Modelos de Machine Learning 28](#_Toc69649677)

[6. Apresentação dos Resultados 38](#_Toc69649678)

[7. Links 53](#_Toc69649679)

[REFERÊNCIAS 54](#_Toc69649680)

# 1. Introdução

O trabalho visa à criação de playlists no spotify utilizando as características musicais (áudio features) presentes. Para isso, o estudo foi estruturado em seis partes. Na introdução, será apresentada a contextualização do assunto tratado e o problema proposto. Na sequência, serão apresentadas as informações acerca da coleta dos dados (Capítulo 2) e os passos realizados para o tratamento inicial, consistente em sua limpeza e estruturação (Capítulo 3). Passo seguinte, serão efetuadas a análise e a exploração desses dados, com vistas à obtenção de informações estatísticas relevantes (Capítulo 4). Como o trabalho gira em torno de análise de dados musicais, o Capítulo 5 – Aplicação de Modelos de *Machine Learning* – abrangerá processamento de dados, bem como aplicação de algoritmos para agrupamento dos dados. Por fim, no Capítulo 6 serão apresentados os resultados obtidos.

## 1.1. Contextualização

Em pleno século 21 seria loucura dizer que o que temos de mais valioso, atualmente, são as informações? E já digo que a minha pergunta na verdade faz muito sentido. Um século atrás, o recurso em questão era o petróleo. Só para termos uma idéia real do que estou dizendo, temos as cincos corporações mais valiosas do mundo que são a [Google](https://brasil.elpais.com/tag/google/a), Apple, Facebook, Amazon e a Microsoft. Elas valem até 3 trilhões de dólares (cerca de 10 trilhões de reais) na Bolsa, ou seja, pouco mais do que o PIB do [Reino Unido](https://brasil.elpais.com/tag/reino_unido/a). Mas confesso que apesar de ser bastante interessante, esse não é o foco do meu projeto, na verdade era só um gancho para falar da importância da informação e do quanto as informações se tornaram valiosas atualmente. Então quer dizer que esse projeto não aborda a análise das informações? Bom na verdade sim, porém digamos que seja abordando algo que para maioria das pessoas está presente na maior parte do tempo e nesse caso estou falando da música.

No mercado da música, a internet trouxe a música digital, e como sabemos a música desempenha um grande papel na vida de muitas pessoas, nos dias de hoje somos surpreendidos com novos sons, estilos, bandas, tendências e experiências musicais, que nos provocam sensações e experiências únicas e pessoais, mas não é de hoje que o ser humano aprecia essa experiência. Com relação ao mercado da música digital, percebemos que as plataformas de streaming trazem para o consumidor uma forma mais prática para se consumir música, tornando-se um novo hábito. Os serviços de streaming vêm dominando o mercado fonográfico. De acordo com dados da Federação Internacional da Industria Fonográfica (IFPI), a indústria da música gravada registrou a marca de US$ 20,2 bilhões em 2019, um número 8,2% maior do que no ano anterior. É a primeira vez que a indústria alcança a marca dos US$ 20 bilhões desde 2005. Este é o quinto ano seguido de crescimento. E como o [Spotify](http://www.techtudo.com.br/tudo-sobre/spotify.html)foi eleito como um dos [aplicativos, para ouvir música, mais populares do momento](https://www.techtudo.com.br/listas/2018/08/como-funciona-o-spotify-entenda-o-aplicativo-de-musica.ghtml) meu projeto será feito utilizando essa plataforma.

## 1.2. O problema proposto

Nesse estudo abordaremos a análise de dados musicais, recomendação e criação de playlists. Nosso foco será o Spotify.

Spotify é um aplicativo de streaming de música. Essa plataforma é uma das mais usadas atualmente permite que os assinantes acessem cerca de 35 milhões de músicas a partir de diversos modos de navegação.

Em tradução livre, a palavra stream significa corrente, e o uso da palavra streaming poderia ser definido como corrente contínua (se pensarmos pelo radical da palavra), que é exatamente o que esta tecnologia permite: a transmissão de dados através da internet, de forma contínua.

Fundado por Daniel Ek e Martin Lorentzon em 2008 na Suécia, e disponível no Brasil desde 2014. O Spotify está presente em 65 países e disponibiliza no seu player mais de 35 milhões de músicas, todas com seus direitos autorais respeitados: possui acordos com as três grandes gravadoras – Universal, Sony e Warner. O player diz já ter gasto de direitos autorais algo em torno de 5 bilhões de dólares. Como um serviço em streaming, o Spotify funciona a partir de uma memória musical em nuvem, disponibilizando faixas para os seus usuários consumirem online.

Este projeto é composto resumidamente pelas etapas: coleta de dados, análises e apresentação. Os dados são as playlists públicas criadas pelos usuários da plataforma. Para isso, utilizamos a API do Spotify e a biblioteca Spotipy, que localizam os Ids de todas a playlist do usuário e com essa informação consegue coletar todas as músicas de cada playlist. Os dados capturados são armazenados em um Dataframe.

Os dados são acessados e passam por tratamentos e análises. Por fim, as análises são apresentadas em forma de gráficos para visualização imediata

As análises feitas são do tipo quantitativa, onde identificamos as informações numéricas como quantidades gerais, contagem de valores específicos e taxas de proporção. E também do tipo qualitativa, em que classificamos através de um modelo de aprendizagem de máquina, o sentimento presente nas músicas, no caso iremos utilizar as áudio features padrões na qual o spotify utiliza para podermos realizar as recomendações e a criação de playlist. Iremos detalhar essas áudio features mais à frente.

O projeto tem foco em um usuário do Spotify, com isso a coleta será feita abordando todas as músicas do usuário, onde iremos utilizar para realizar a coleta e a análise das músicas para a recomendação e criação de playlists.

## Tecnologias utilizadas

[Python](https://www.python.org/): Linguagem de programação. Bem como diversas bibliotecas e pacotes compatíveis ou criados para essa linguagem:

[Pandas:](https://www.anaconda.com/distribution/) Pacote para manipulação de dados em formato dataframe;

Sklearn: Biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python;

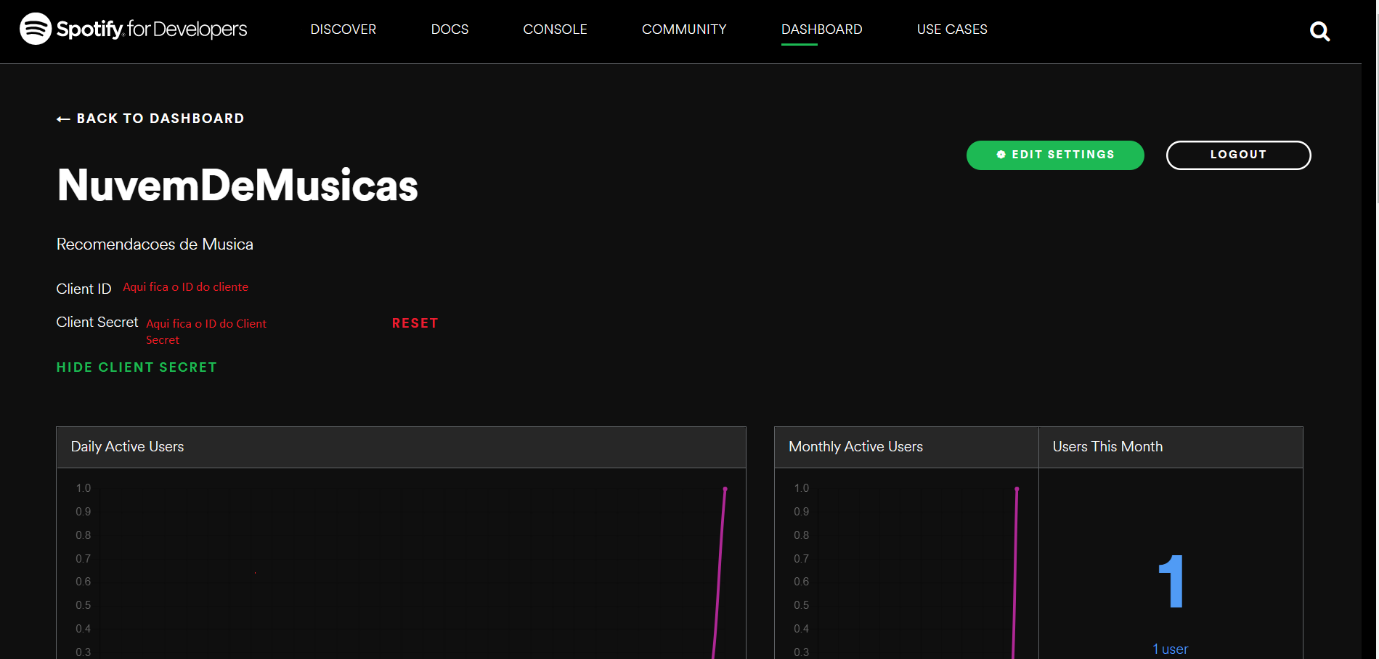
[Spotipy](https://tweepy.readthedocs.io/en/latest/): Biblioteca Open-source para acessar a API do Spotify;

[Jupyter notebook:](https://www.anaconda.com/distribution/) IDE para linguagem python

[Plotly:](https://plot.ly/) Framework para criação de gráficos e dashboards;

# 2. Coleta de Dados

Neste projeto, utilizamos dados obtidos da própria plataforma Spotify. A API permite obter dados ou recomendações do usuário, criar ou modificar listas de reprodução e muito mais. Para usá-lo, precisamos passar por uma configuração inicial com algumas etapas que são fazer login/registrar como desenvolvedor no [site](https://developer.spotify.com/dashboard/login) e criar um aplicativo. Lembrando que após a criação do aplicativo a plataforma automaticamente gera duas chaves que são a Client ID e a Secret Client. Com essas chaves será possível realizar a autenticação da API. (a Figura 1 mostra as chaves geradas após a criação da aplicação)



*Figura 1 - Aplicativo My Dashboard*

O acesso as playlists dos usuários se dão utilizando as chaves da API. O Spotify disponibiliza um serviço no qual diversas métricas do seu conteúdo são disponibilizadas. Com um cadastro e em posse de suas credenciais você tem acesso. Como suporte utilizei a biblioteca Spotipy, que nada mais é do que uma biblioteca Python leve para a [API da Web](https://developer.spotify.com/web-api/) do [Spotify](https://developer.spotify.com/web-api/).

A plataforma suporta dois fluxos de autorização que são o Fluxo do código de autorização e o fluxo de credenciais do cliente. Para o nosso projeto optamos pelo fluxo do código de autorização, pois esse método é adequado para aplicativos de longa execução nos quais o usuário efetua login apenas uma vez e é fornecido um token de acesso que pode ser atualizado.

Esse método exige que seja adicionado um [URI de redirecionamento](https://spotipy.readthedocs.io/en/2.17.1/#redirect-uri)ao seu aplicativo em [My Dashboard](https://developer.spotify.com/dashboard/applications) e também temos que informar o escopo. Os escopos fornecem aos usuários, que usam aplicativos de terceiros, à confiança de que apenas as informações que eles escolherem compartilhar serão compartilhadas e nada mais. Iremos abordar somente os escopos utilizados no projeto, caso tenha interesse e queira abordar ainda mais basta acessar o [site](https://developer.spotify.com/documentation/general/guides/scopes/), segue abaixo a descrição de cada escopo utilizado no projeto:

**Autorizacão dos Escopos:**

* playlist-read-private – Acesso de leitura para as playlists privadas do usuário;
* playlist-read-collaborative – Acesso `a playlists colaborativas do usuário atual,

ou de um determinado usuário;

* user-top-read – Acesso de leitura aos principais artistas e tracks de um

usuário;

* playlist-modify-public – Acesso de gravação `as playlists públicas de um

usuário;

* playlist-modify-private – Acesso de gravação `as playlists privadas de um

usuário;

* user-follow-read – Acesso de leitura `a tracks de artistas e outros usuários

que o usuário segue;

* user-follow-modify – Acesso de gravação/exclusão `a tracks de artistas

e outros usuários que o usuário segue;

* user-read-recently-played – Acesso de leitura `as tracks reproduzidas

recentemente por um usuário;

* user-read-currently-playing – Acesso de leitura ao conteúdo em reprodução

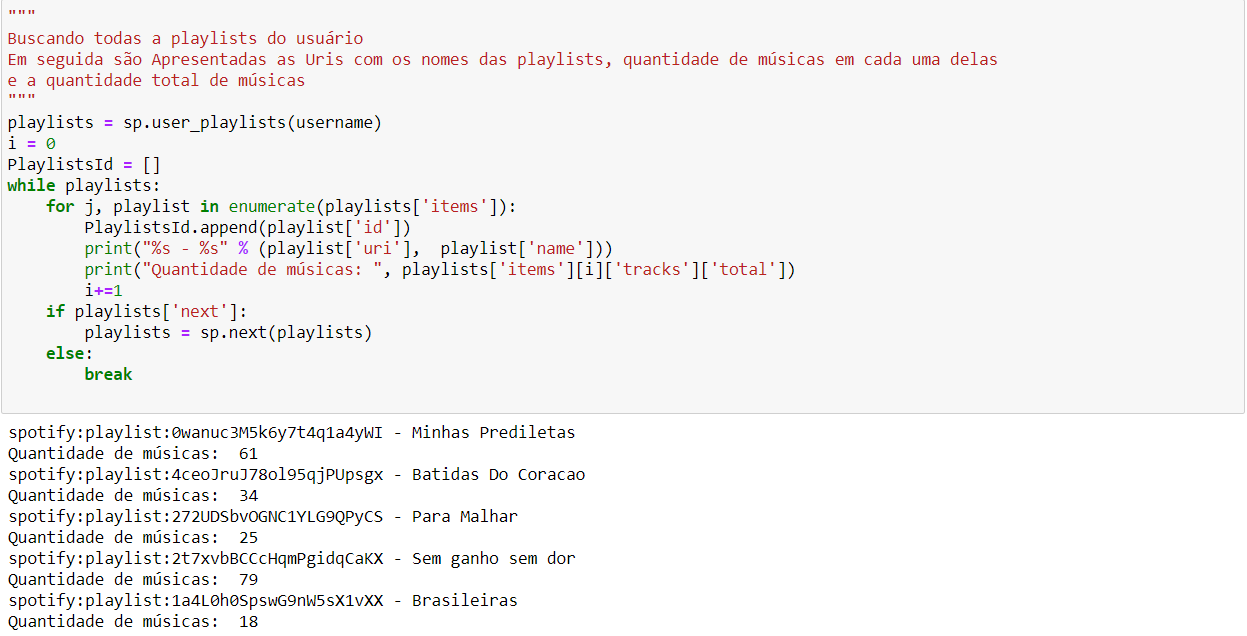
de um usuário.

Para apoiar o fluxo de código de autorização do cliente, o *Spotipy* fornece uma classe SpotifyOAuth que pode ser usada para autenticar solicitações. Com essa classe conseguimos gerar um token de acesso. (Figura 1.1)



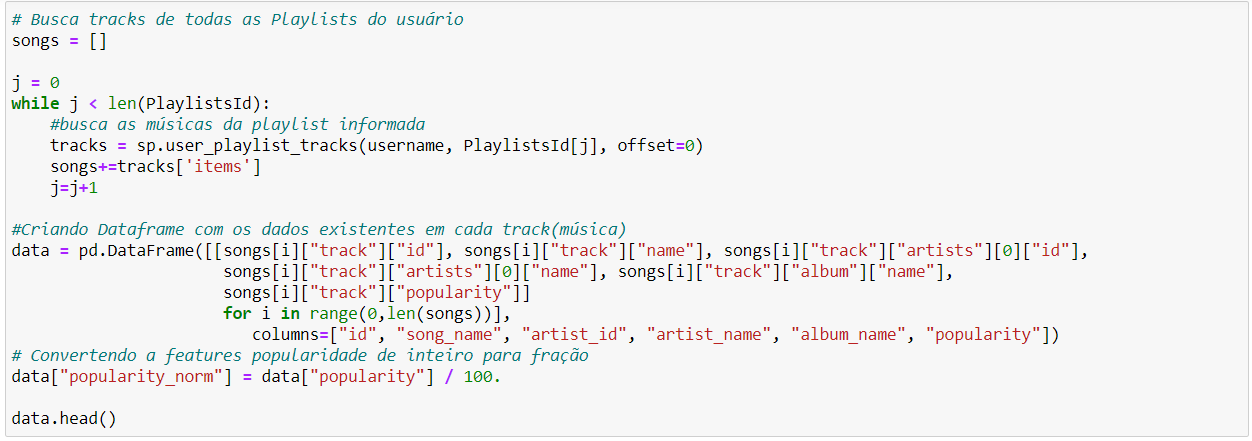
*Figura 1.1 - Exemplo classe SpotifyOAuth*

Os tokens são atualizados automaticamente e armazenados por padrão na pasta principal do projeto. Neste ponto, temos um objeto Spotify para coletar dados online. Com esse objeto conseguimos acesso a algumas funções para coletas de dados. A imagem abaixo mostra as extrações dos IDs das playlists do usuário ‘betatunes’. Para essa extração foi utilizada a função playlists = sp.user\_playlists(username). (Figura 1.2)



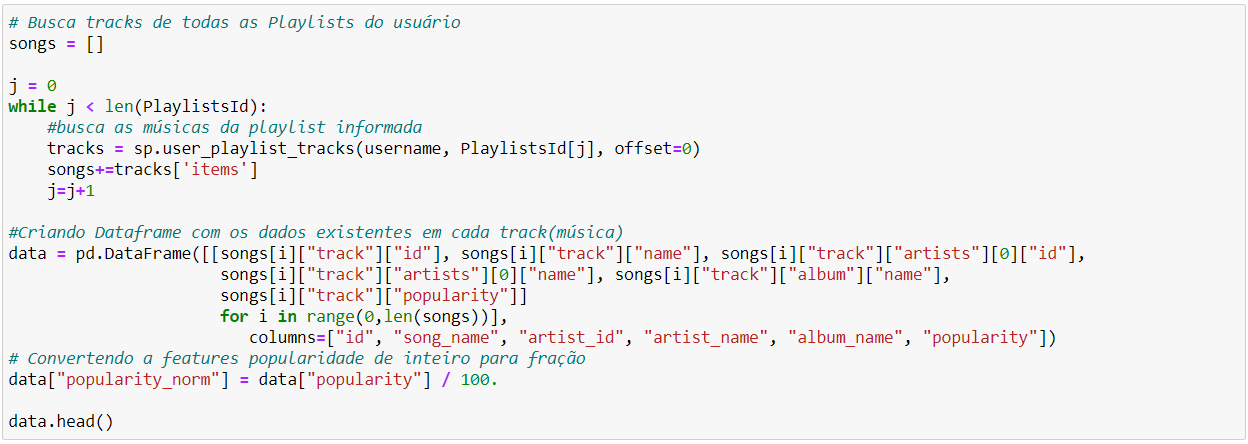
*Figura 1.2 - Coleta dos Ids da playlists*

Com os IDs da playlists conseguimos realizar a coleta de todas as músicas pertencentes ao usuário. (Figura 1.3)



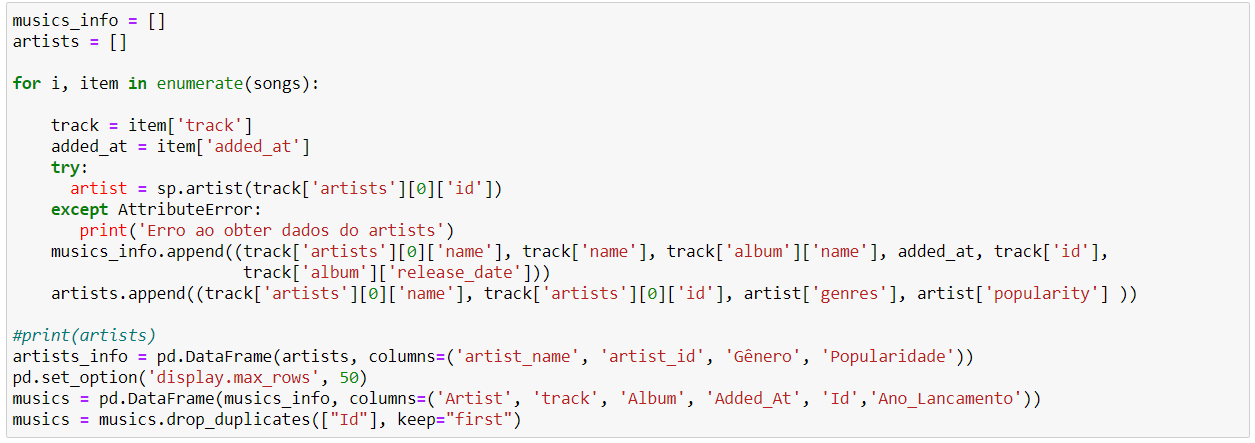
*Figura 1.3 - Coleta das músicas pertencentes ao usuário*

Após a coleta das músicas gravamos as mesmas em um DataFrame, que nada mais é do que uma estrutura de dados bidimensional, como uma matriz bidimensional ou uma tabela com linhas e colunas. Essa estrutura pertence a biblioteca Pandas. O Pandas DataFrame facilita a manipulação de seus dados, desde a seleção ou substituição de colunas e índices até a remodelagem dos dados. (Figura 1.4)



*Figura 1.4 - Dados das músicas gravados na variável songs*

Após a coleta criamos dois DataFrames, uma para guardar os dados das músicas e o outro para guardar os dados dos artistas. (Figura 1.5)

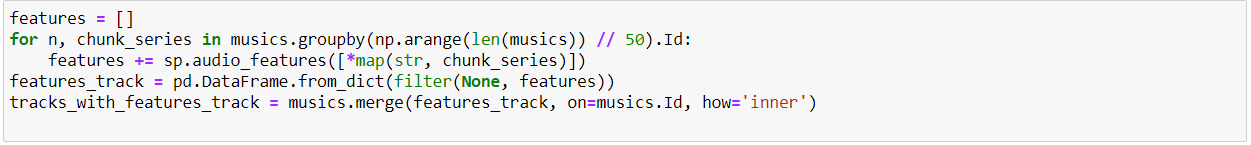


*Figura 1.5 - Coleta das músicas*

Agora chegamos no ponto principal, onde iremos coletar as informações que serão utilizadas para realizarmos os agrupamentos musicais e com isso realizar as recomendações e as criações das playlists que são o ponto principal do projeto. Os dados do nosso interesse são as áudio features.

Como sabemos, o Spotify é uma das plataformas de streaming de áudio mais populares em todo o mundo. Possui uma API para que os desenvolvedores explorem seu banco de dados de música e obtenham insights sobre nossos hábitos de escuta.

Criei um DataFrame adicionando as colunas relacionadas aos recursos de áudio. (Figura 1.6)



*Figura 1.6 - DataFrame adicionando as áudio features*

Você pode encontrar abaixo a explicação de cada recurso (passado / cópia do site do Spotify). (Figura 1.7)

**- Acusticness (tipo float):** Índice que mede se a música é acústica ou não. Valores entre 1 e 0 onde um valor 1 é uma música totalmente acústica e um valor 0, ou próximo, diz que a música não tem elementos de música acústica.

**- Danceability (tipo float):** Danceability descreve como uma faixa é adequada para a dança baseada em uma combinação de elementos musicais, incluindo tempo, estabilidade de ritmo, força de batida e regularidade geral. Um valor de 0.0 é menos dançante e 1.0 é mais dançante;

**- Duration\_ms (tipo int):** A duração da faixa em milissegundos;

**- Energy (tipo float):** Energia é uma medida de 0,0 a 1,0 e representa uma medida perceptual de intensidade e atividade. Normalmente, faixas energéticas são rápidas e barulhentas;

**- Instrumentalness (tipo float):** Prevê se uma faixa não contém vocais. Os sons “Ooh” e “aah” são tratados como instrumentais neste contexto. Faixas de rap ou palavra falada são claramente “vocais”. Quanto mais próximo o valor de instrumental for de 1,0, maior a probabilidade de a faixa não conter conteúdo vocal;

**- Key (tipo int):** O tom na qual a faixa se encontra. Os inteiros mapeiam para campos de notas usando a notação de classe de “pitch” padrão. Por exemplo, 0 = C, 1 = C] / D [, 2 = D e assim por diante;

**- Liveness (tipo float):** Detecta a presença de um público na gravação. Valores mais altos de atividade representam uma probabilidade maior de que a faixa foi executada ao vivo. Um valor acima de 0,8 fornece uma forte probabilidade de que a faixa esteja ativa;

**- Loudness (tipo float):** O volume total de uma faixa em decibéis (dB). Os valores de sonoridade são calculados ao longo de toda a faixa e são úteis para comparar a intensidade relativa das faixas. Valores típicos variam entre -60 e 0 db;

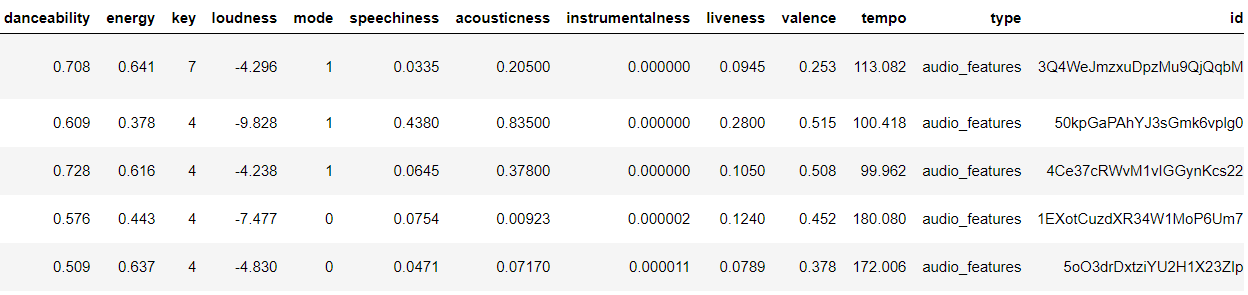
**- Mode (tipo int):** Mode indica a modalidade (maior ou menor) de uma faixa, o tipo de escala a partir da qual seu conteúdo melódico é derivado. Maior é representado por 1 e menor é 0;

**- Speechiness (tipo float):** A fonação detecta a presença de palavras faladas em uma faixa. Quanto mais exclusivamente discursiva a gravação (por exemplo, talk show, livro de áudio, poesia), quanto mais próximo de 1,0 o valor do atributo. Valores acima de 0,66 descrevem faixas que provavelmente são feitas inteiramente de palavras faladas. Valores entre 0,33 e 0,66 descrevem faixas que podem conter música e fala, seja em seções ou em camadas, incluindo casos como música rap. Valores abaixo de 0,33 provavelmente representam músicas e outras faixas não relacionadas à fala;

**- Tempo (tipo float):** O tempo total estimado de uma faixa em batidas por minuto (BPM). Na terminologia musical, tempo é a velocidade ou ritmo de uma determinada peça e deriva diretamente da duração média da batida;

**- Time\_signature (tipo int):** Um tempo estimado global de assinatura de uma faixa. A assinatura de tempo (metro) é uma convenção de notação para especi­car quantas batidas estão em cada barra (ou medida);

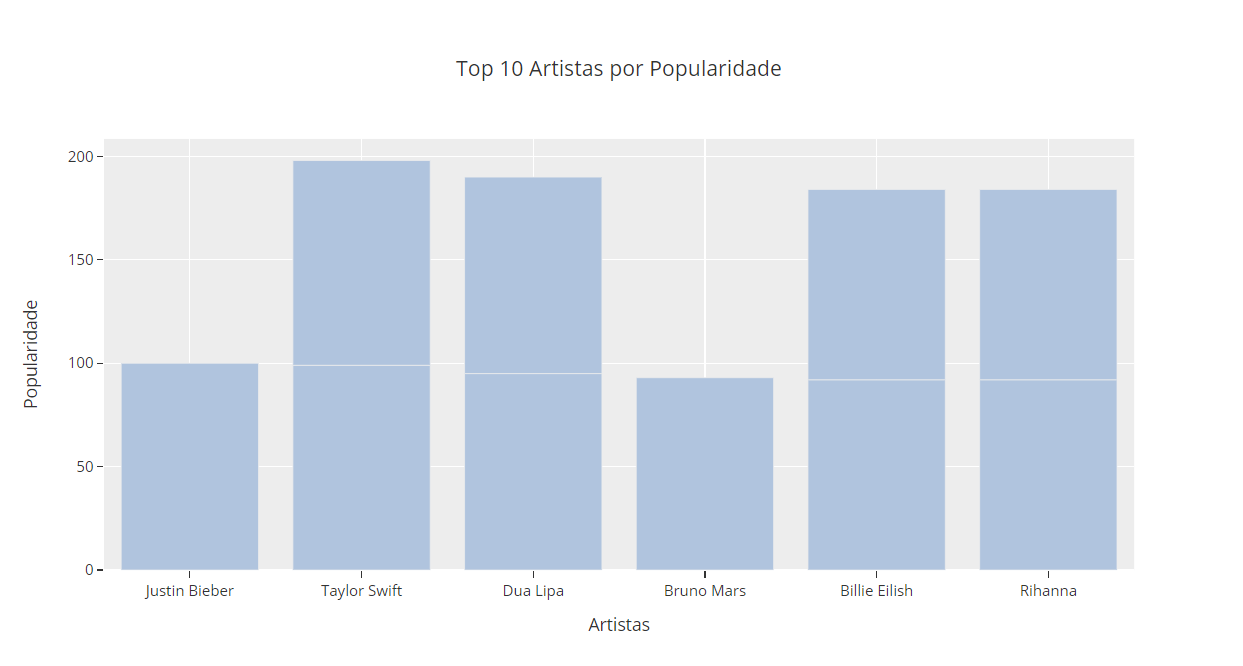
**- Valence (tipo float):** Uma medida de 0,0 a 1,0 descrevendo a positividade musical transmitida por uma faixa. Faixas com alta valência soam mais positivas (por exemplo, felizes, alegres, eufóricas), enquanto as faixas com baixa valência soam mais negativas (por exemplo, triste, deprimido, zangado).



*Figura 1.7 - DataFrame criada com as áudio features*

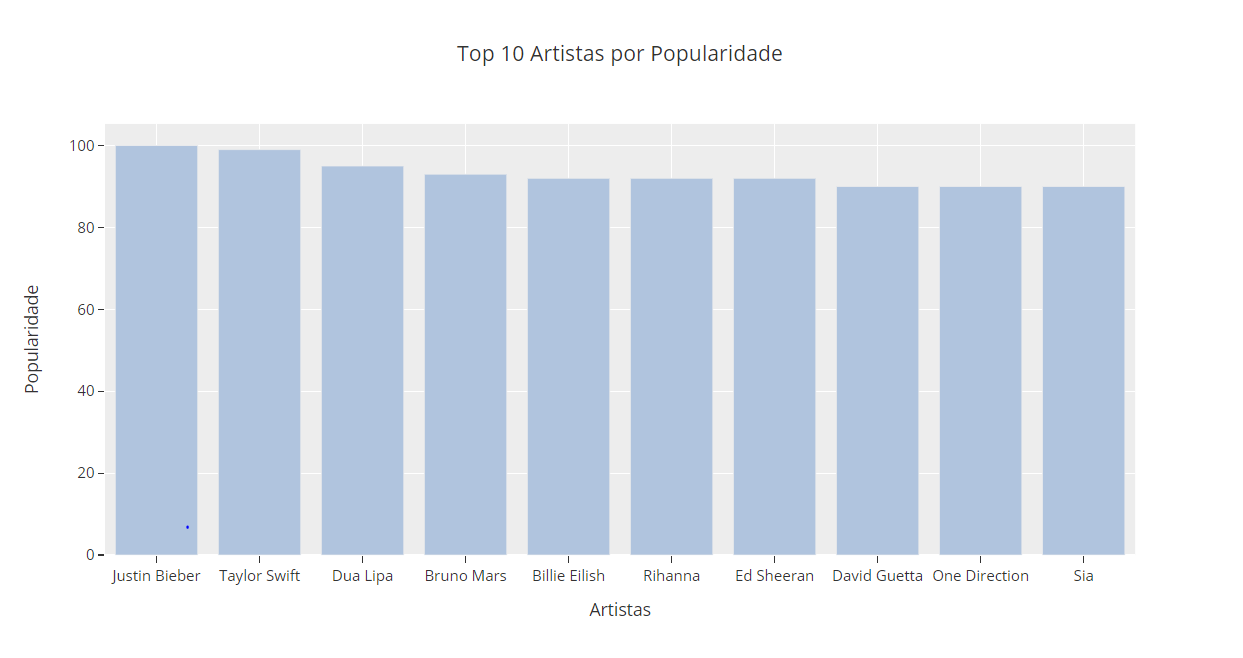
# 3. Tratamento de Dados

Felizmente os dados do Spotify possuem uma boa estrutura, praticamente pronta para iniciar a análise exploratória. Os endpoints da API Web do Spotify retornam metadados no formato JSON, fornece acesso a dados relacionados ao usuário, como listas de reprodução e músicas que o usuário salva na biblioteca na qual chamamos de playlists. Justamente por essa facilidade tivemos que realizar poucos tratamentos nos dados. O primeiro no caso foi o tratamento de dados duplicados. Para ficar mais claro o problema, irei ilustrar o mesmo com um exemplo. Ao listar os Top 10 artistas por popularidade, estavam sendo retornados apenas seis artistas ao invés de dez. (Figura 2)



*Figura 2. Resultado sem a tratativa*

Depois de analisar os dados retornados no Dataframe foi verificado que alguns dados estavam duplicados, ocasionando erro na apresentação do resultado. Quando inseri a tratativa, passou a retornar os valores corretamente. (Figura 2.1)



*Figura 2.1 - Resultado com os dados tratados*

Para o não comprometimento de resultados optamos por realizar a tratativa nas áudio features coletadas na API, como apresentamos anteriormente a maior parte das features trabalham com valores entre 0 e 1.

Vamos colocar um exemplo para que fique mais claro. Sabemos que a feature ‘valence’ utiliza valores entre 0 e 1, sendo que 1 é o valor máximo e 0 o valor mínimo, já no caso da ‘popularity’ ela fica com o valor máximo 100 e mínimo 0 e como esses dados serão utilizados para a análise dos dados achei que para atingir um melhor resultado, seria correto normalizar esses dados, padronizando todos eles para valores entre 0 e 1. Optamos por não utilizar a feature loudness já que a mesma trabalha com valores entre 0 e -60, então a mesma poderia influenciar de forma negativa na análise e apresentação dos dados.

|  |
| --- |
|  |

# 4. Análise e Exploração dos Dados

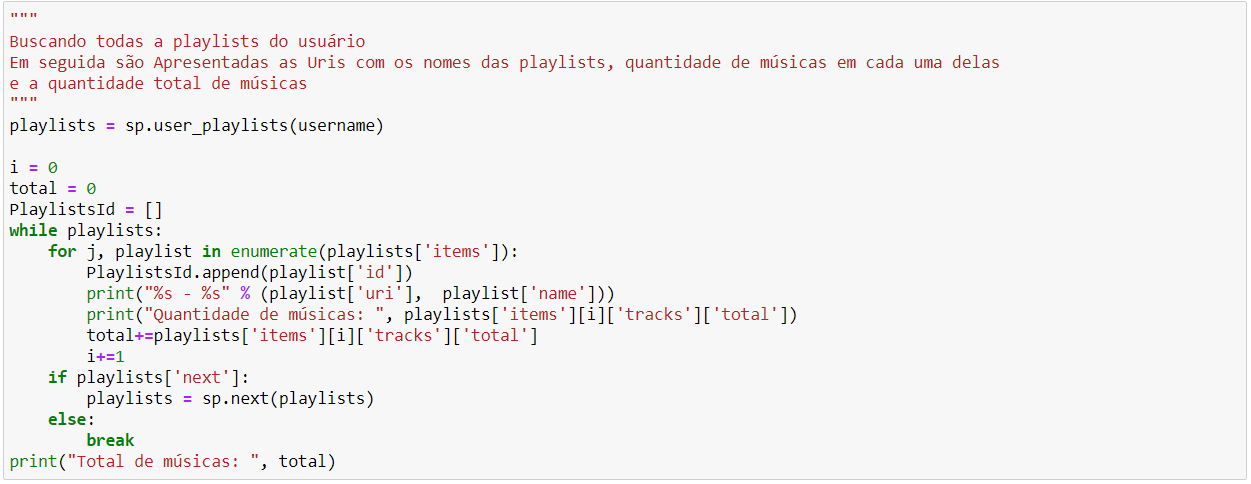
Achei que seria interessante colocar alguns tópicos que nos ajudarão a transformar os dados em informações por meio das análises quantitativas. São eles:

* Informações das Playlists;
* Obtenção das músicas;
* Top 10 artistas por popularidade;
* Top 10 Artistas presentes nas playlists do usuário por quantidade de música;
* Top 10 artistas com maior popularidade;
* Top 20 Músicas por Popularidade;
* Top 10 Gêneros Musicais;
* Top 10 Artistas por Quantidade de Seguidores;
* Top 10 Quantidade de Músicas por Álbum

Nesta etapa é utilizado o pacote Pandas e o pacote Plotly juntamente com algumas outras bibliotecas python como o scikitlearn.

**Informações das Playlists:**

Para a obtenção das informações das playlists do usuário utilizamos a função sp.user\_playlists(), com essa função, disponibilizada pela biblioteca Spotipy, tivemos acesso a uma lista de dados das listas de reprodução (playlists). Com o resultado dessa função conseguimos buscar o URI de cada playlist, o nome e o total de músicas que cada playlists possui. URI do Spotify ( Uniform Resource Indicator) é um link que você pode encontrar no menu Compartilhar de qualquer faixa, playlist, álbum ou perfil do artista do Spotify. Quando você clica em um link que consiste em um URI do Spotify  (em vez de um endereço HTTP), você é levado diretamente para o  aplicativo Spotify, sem ter que passar pela página da web primeiro. (Figura 3)



*Figura 3 – Coleta dos Ids das Playlists*

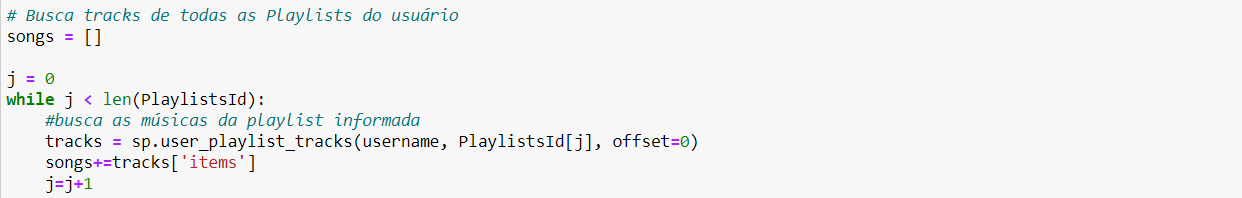
**Obtenção das músicas:**

Para a obtenção das músicas de todas as listas de reprodução do usuário seguimos alguns processos, o primeiro deles, falado anteriormente, foi utilizar a função sp.user\_playlists(), lembrando que isso dá acesso às listas de reprodução que o usuário possui e as listas de reprodução que o usuário está seguindo, mas não temos acesso às listas de reprodução colaborativas pertencentes a outros usuários. Dentre todos os dados que estamos buscando, o que precisamos de fato são os Ids das listas de reprodução. (Figura 3.1)



*Figura 3.1 - Ids das Playlists*

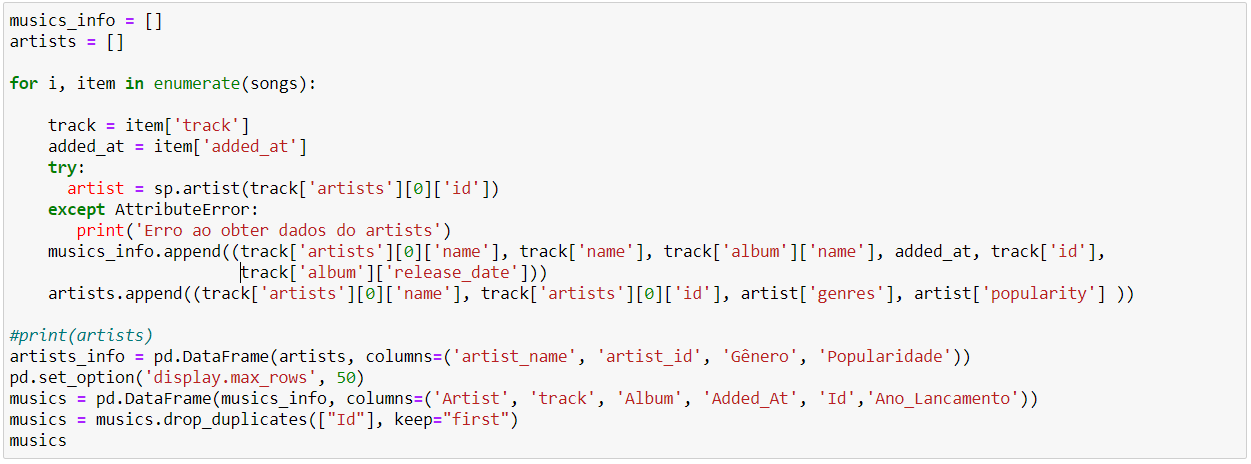
Passando do processo de busca dos Ids chegamos no processo de busca das músicas presentes em cada lista de reprodução e para isso utilizamos a função sp.user\_playlist\_tracks(). Com essa função consigo pegar todas as músicas presentes nas listas de reprodução do usuário. (Figura 3.2)



*Figura 3.2 – Coleta das músicas presentes nas playlists do usuário*

Para a coleta das informações, que serão apresentadas a seguir, além da biblioteca spotipy, foi utilizado as bibliotecas pandas, pyplot e plotly.

Com a obtenção das músicas conseguimos coletar as informações necessárias para guardar em um DataFrame. Para a coleta dos dados dos artistas utilizamos a função sp.artist(artist\_id), para que possamos pegar essas informações precisamos passar como parâmetro o id do artista, abaixo temos o código utilizado desde a chamada da função até a obtenção dos dados guardando em uma variável, chamada artists, do tipo lista. (Figura 3.3)

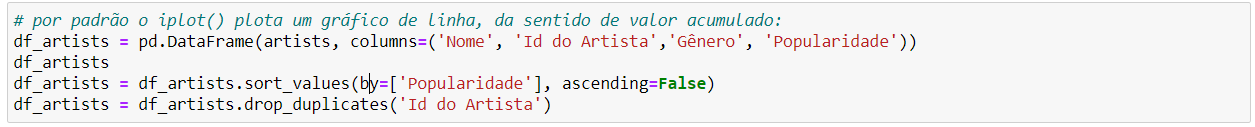


*Figura 3.3 – Criação de DataFrame com as informações das músicas e dos artistas*

Depois que adicionamos os dados dos artistas nessa variável, criamos um DataFrame e jogamos todas as informações nesse DataFrame. E devido a facilidade que o DataFrame nos proporciona podemos utilizar algumas funcionalidades que nos ajudam a listar os artistas com maior popularidade.

O resultado é representado por um gráfico utilizando a biblioteca plotly. Para alcançar as demais informações foram utilizamos praticamente a mesma lógica apresentada no tópico anterior. Portanto iremos apenas colocar o código de classificação dos dados e o código utilizado para a construção dos gráficos. (Figuras 3.4 a 3.19)

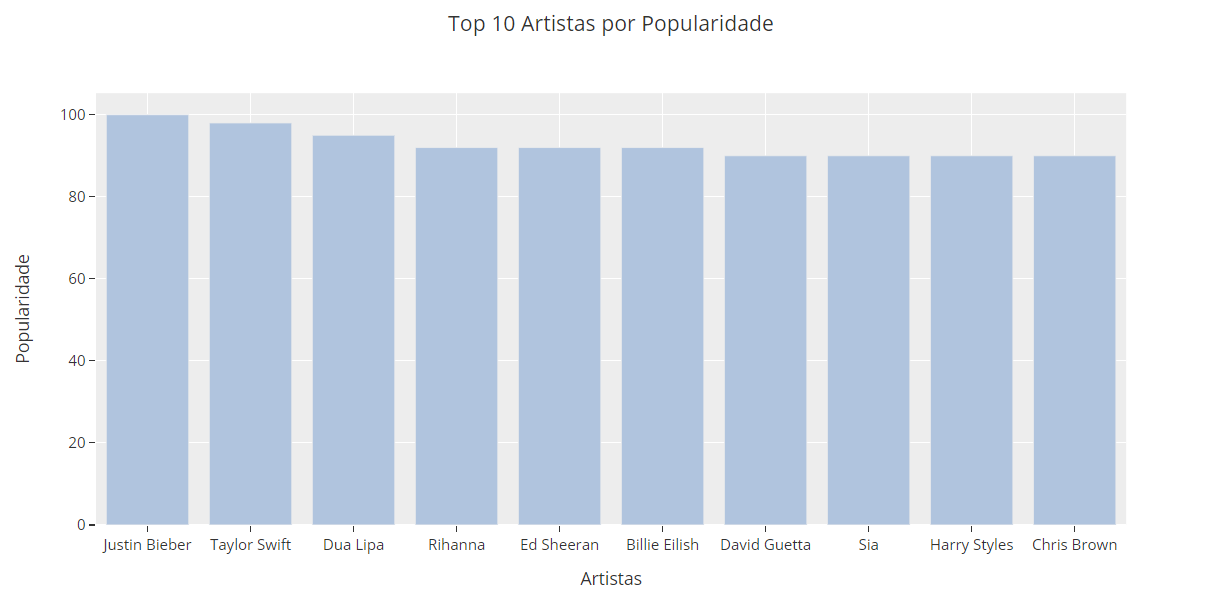
**Top 10 Artistas por popularidade**



*Figura 3.4 – Criação de DataFrame agrupando os artistas por Popularidade*

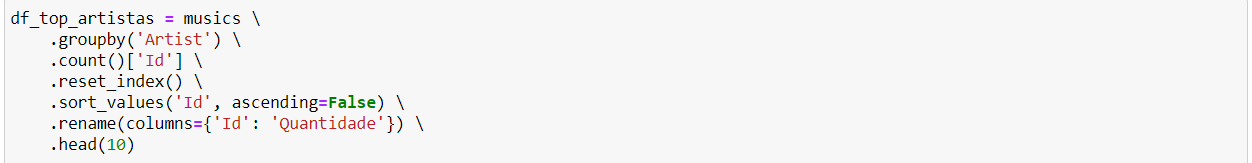


*Figura 3.5 – Construção do gráfico mostrando os Top 10 artistas por popularidade*

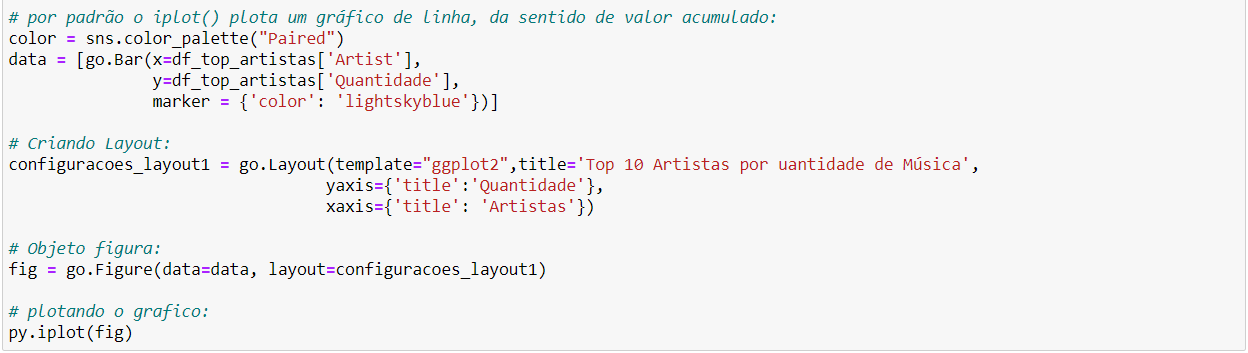


*Figura 3.5 – Gráfico dos Top 10 artistas por Popularidade*

**Top 10 Artistas presentes nas playlists do usuário por quantidade de música**



*Figura 3.6 – Criação de DataFrame agrupando os artistas por Quantidade de música*

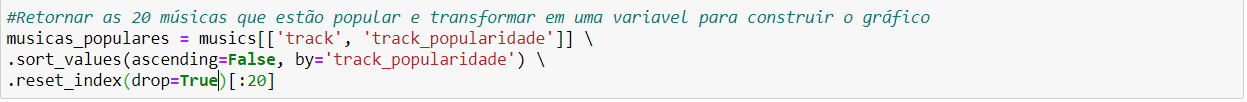


*Figura 3.7 – Construção do gráfico mostrando os Top 10 artistas por quantidade de música*



*Figura 3.8 – Gráfico dos Top 10 artistas por Quantidade de música*

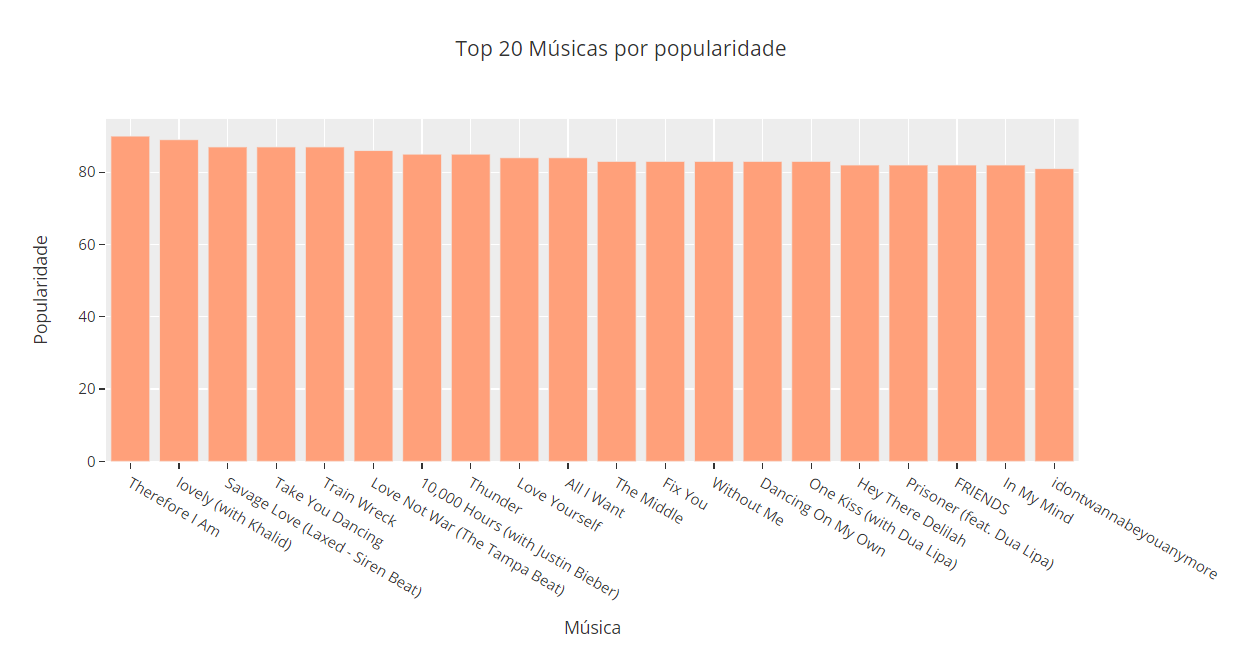
**Top 20 Músicas por Popularidade**



*Figura 3.9 – Criação de DataFrame agrupando as músicas por popularidade*



*Figura 3.10 – Construção do gráfico mostrando as Top 20 músicas por popularidade*



*Figura 3.11 – Gráfico das Top 20 músicas por popularidade*

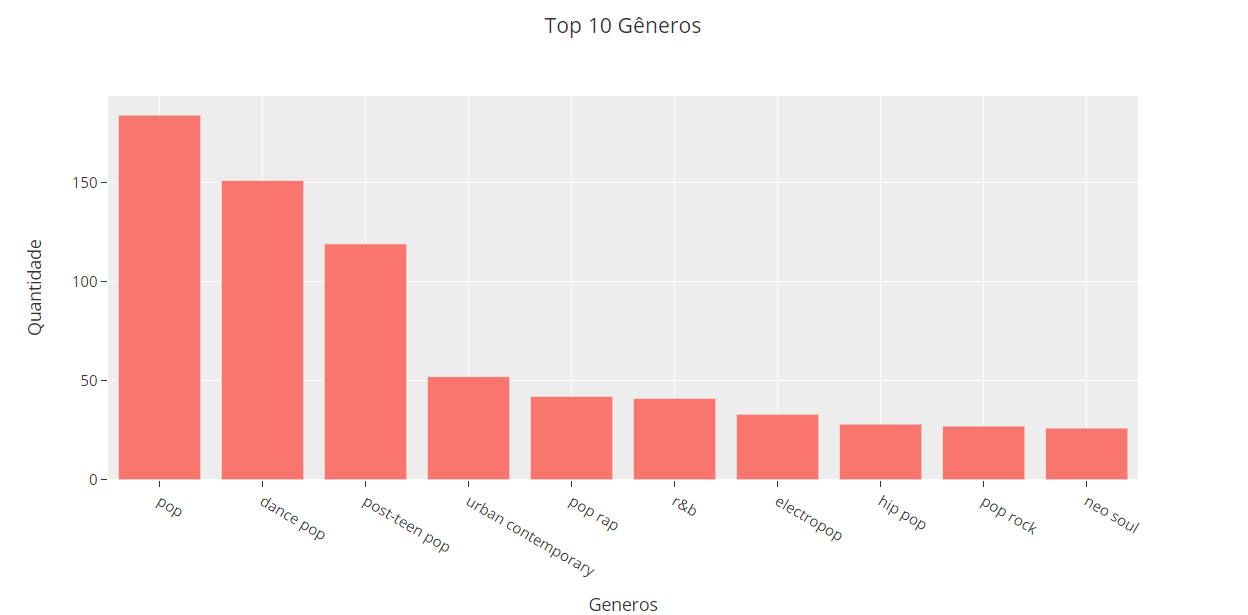
**Top 10 Gêneros Musicais**



*Figura 3.12 – Criação de DataFrame agrupando os Gêneros por quantidade*



*Figura 3.13 – Construção do gráfico mostrando os Top 10 gêneros musicais*

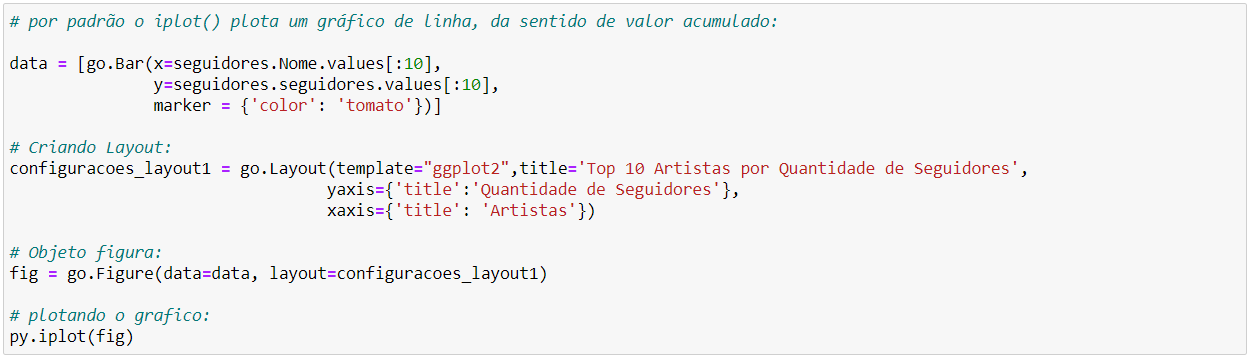


*Figura 3.14 – Gráfico dos Top 10 Gêneros musicais*

**Top 10 Artistas por Quantidade de Seguidores**



*Figura 3.15 – Criação de DataFrame agrupando os Artistas por quantidade de seguidores*

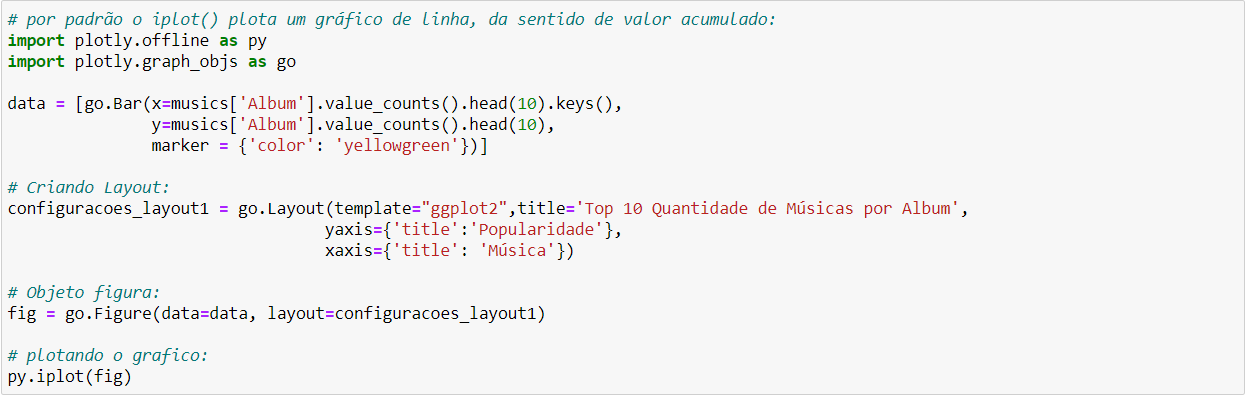


*Figura 3.16 – Construção do gráfico mostrando os Top 10 artistas por seguidores*

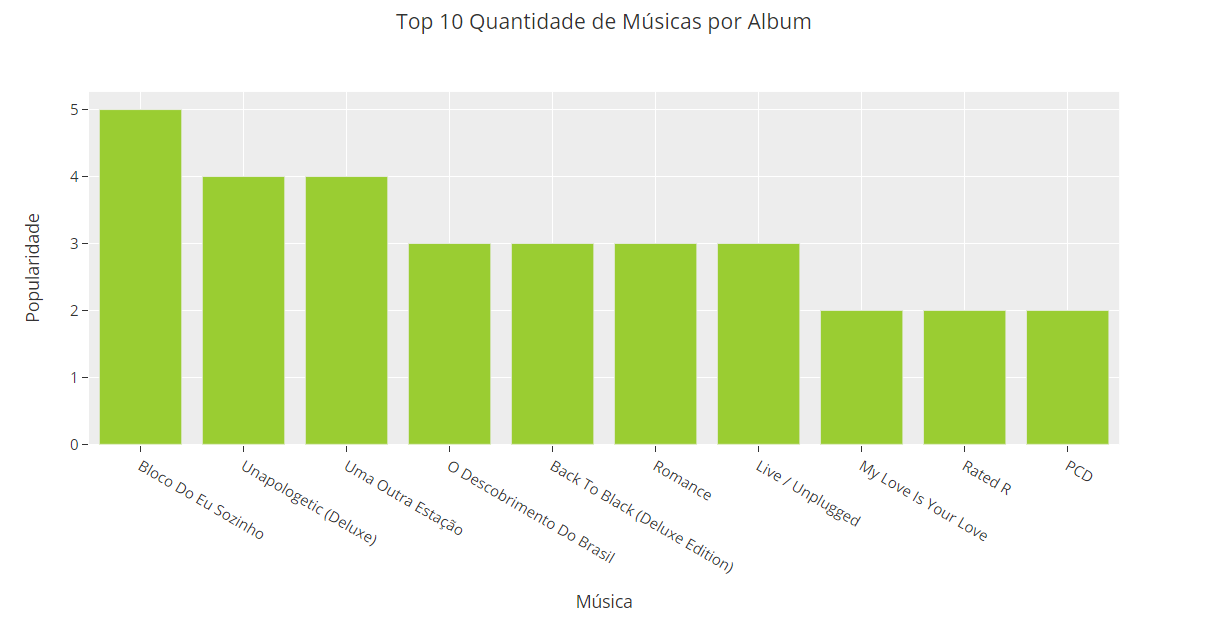


*Figura 3.17 – Gráfico dos Top 10 Artistas por Quantidade de seguidores*

**Top 10 Quantidade de Músicas por Álbum**



*Figura 3.18 – Construção do gráfico mostrando as Top 10 Quantidades de Música por Álbum*



*Figura 3.19 – Gráfico dos Top 10 Quantidades de Música por Álbum*

# 5. Criação de Modelos de Machine Learning

A proposta desse projeto é a utilização dos dados do spotify para a criação de playlists. No caso iremos coletar os dados, analisar as features das músicas e com essas features iremos agrupar as informações e dessas informações agrupadas iremos criar as playlists. E para isso iremos utilizar modelos de machine Learning.

Machine Learning é uma tecnologia que permite aos computadores aprender a partir dos dados de entrada e construir / treinar um modelo preditivo sem programação explícita.

A biblioteca utilizada é o Scikit. Sklearn ou scikit-learn e é de longe uma das bibliotecas de código aberto mais úteis em Python. É eficiente para modelagem estatística e machine learning. Algumas dessas ferramentas incluem regressão, classificação, redução de dimensionalidade e agrupamento.

A biblioteca scikit-learn foi escrita principalmente em Python e baseada em SciPy, NumPy e Matplotlib. Usa uma interface Python unificada e consistente para implementar vários algoritmos de pré-processamento, aprendizado de máquina, visualização e validação cruzada.

Nesse projeto precisei categorizar os pontos de dados em clusters, e em seguida, usá-los para reunir esses pontos de dados, que no nosso caso são músicas, em uma sequência de músicas, que se tornará nossa lista de reprodução.

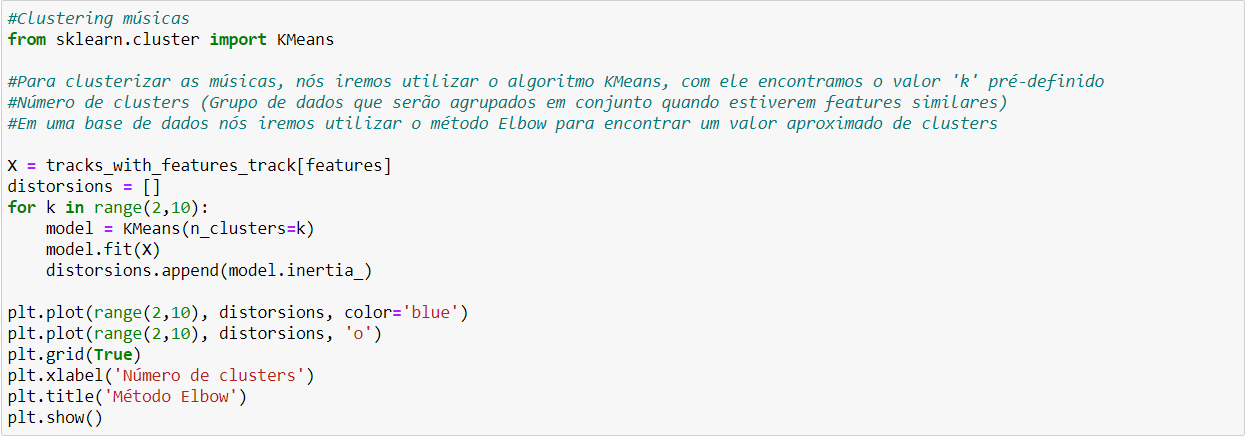
Clustering é uma técnica de ML que trabalha com agrupamento dos dados. Depois que é feita a coleta de um conjunto de dados, é possível utilizar um algoritmo de agrupamento para classificar esses dados em grupos específicos. Os dados serão agrupados de acordo com a semelhança de suas propriedades e recursos.

K-Means é um algoritmo de clusterização (ou agrupamento) disponível na biblioteca Scikit-Learn. É um algoritmo de aprendizado não supervisionado (ou seja, que não precisa de inputs de confirmação externos) que avalia e clusteriza os dados de acordo com suas características.

De modo a agrupar as músicas de acordo com a similaridade entre elas, o algoritmo usado foi o K-Means, o qual encontra e divide as instâncias em um número predefinido de clusters. O K-Means foi escolhido pelo fato de ter se ajustado melhor ao conjunto de dados, levando em consideração outros algoritmos de Clustering.

Basicamente, esse algoritmo define centros (de acordo com a quantidade “k” de clusters) e, para cada uma das instâncias, verifica qual o centróide mais próximo e calcula a média com base nesses agrupamentos.

Apesar de não ser o mais eficiente, o “Elbow Method” foi mais adequado para a situação e, dessa forma, foi usado para a definição do número de clusters. (Figura 4)

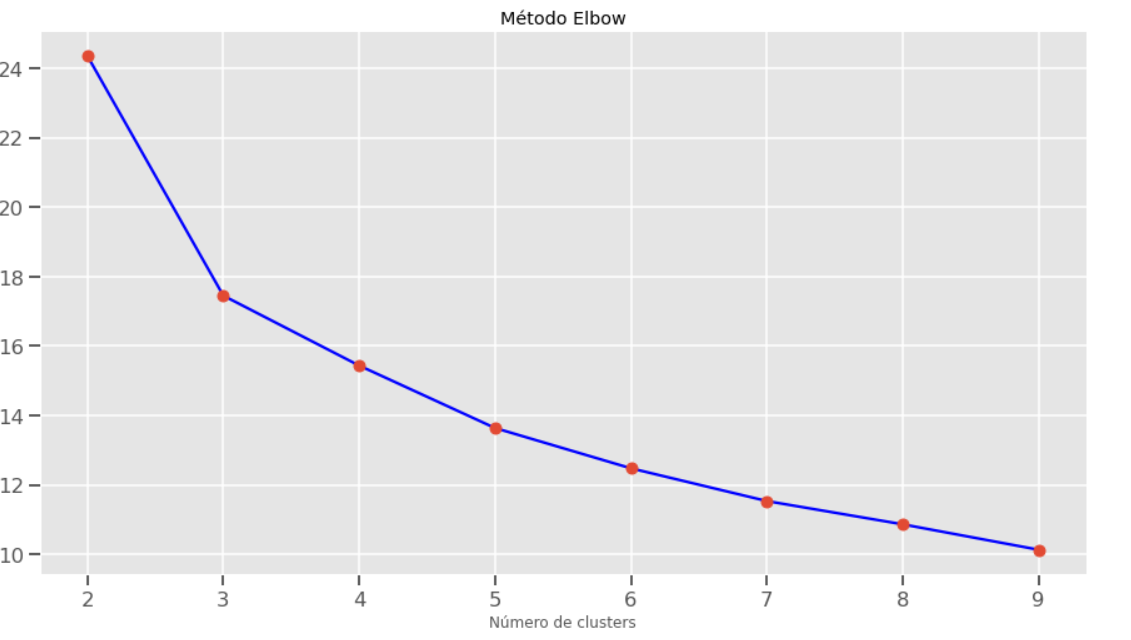


*Figura 4 – Método utilizado para verificar a melhor quantidade de Clusters para realizar o agrupamento*

*das músicas*

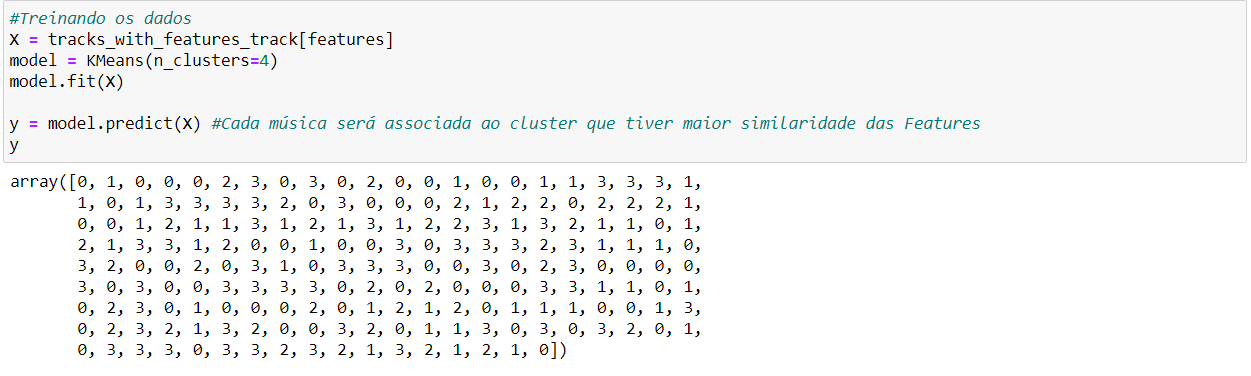
Basicamente, calculamos diferentes cenários para diferentes números de clusters e, em seguida, os plotamos em uma linha.

Para determinar o número de clusters, é necessário selecionar o valor de “k” no ponto após o qual a distorção começa a diminuir de maneira linear. Analisando a imagem abaixo, é possível perceber que isso ocorre quando k=4. Portanto, trabalharemos com 4 clusters. (Figura 4.1)



*Figura 4.1 – Gráfico com o número de clusters*

Usando o K-Means para treinar o modelo e prever o cluster de cada uma das músicas, obtemos os resultados apresentados abaixo. (Figura 4.2)

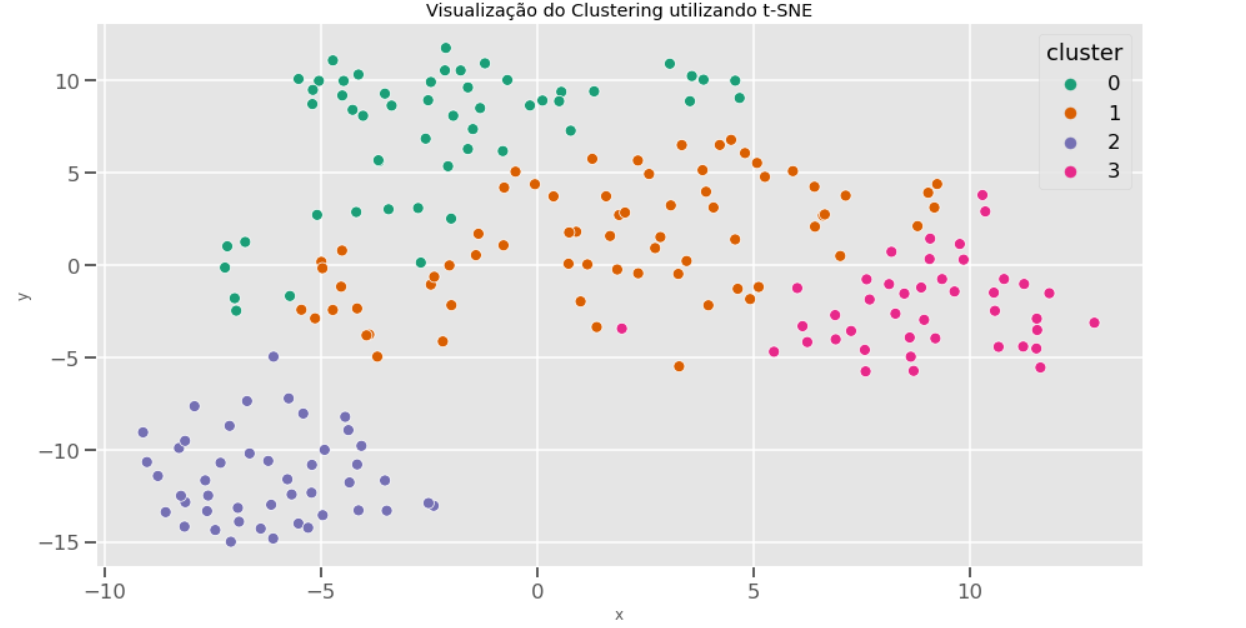


*Figura 4.2 – Prevendo o cluster de cada música*

Pelo fato de trabalharmos com 6 features, e considerando que cada feature representaria uma dimensão, não seria possível visualizar esses clusters. Para visualizar os dados agrupados, usaremos dois algoritmos: PCA e t-SNE. Estes são responsáveis por aplicar técnicas de redução de dimensionalidade e, dessa forma, possibilitar a visualização dos clusters em 2d. O t-SNE, apresentado abaixo, procura preservar pequenas distâncias aos pares. (Figura 4.3 e 4.4)

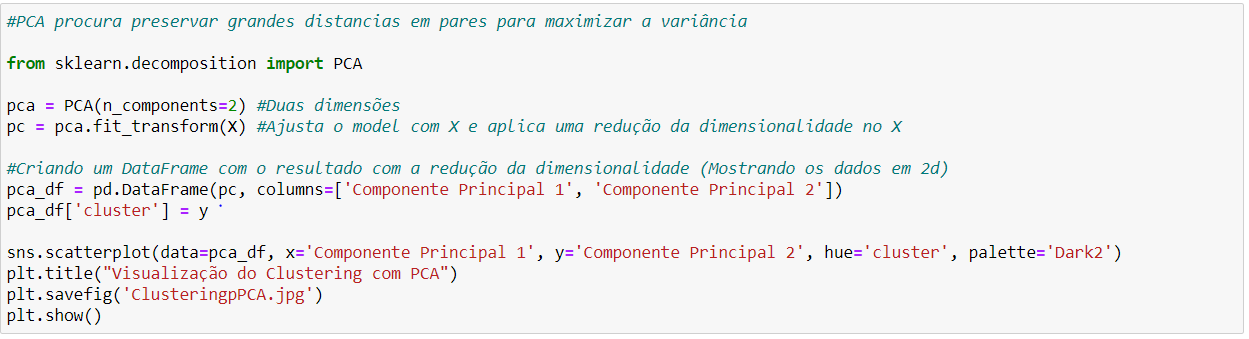


*Figura 4.3 – Algoritmo t-SNE*

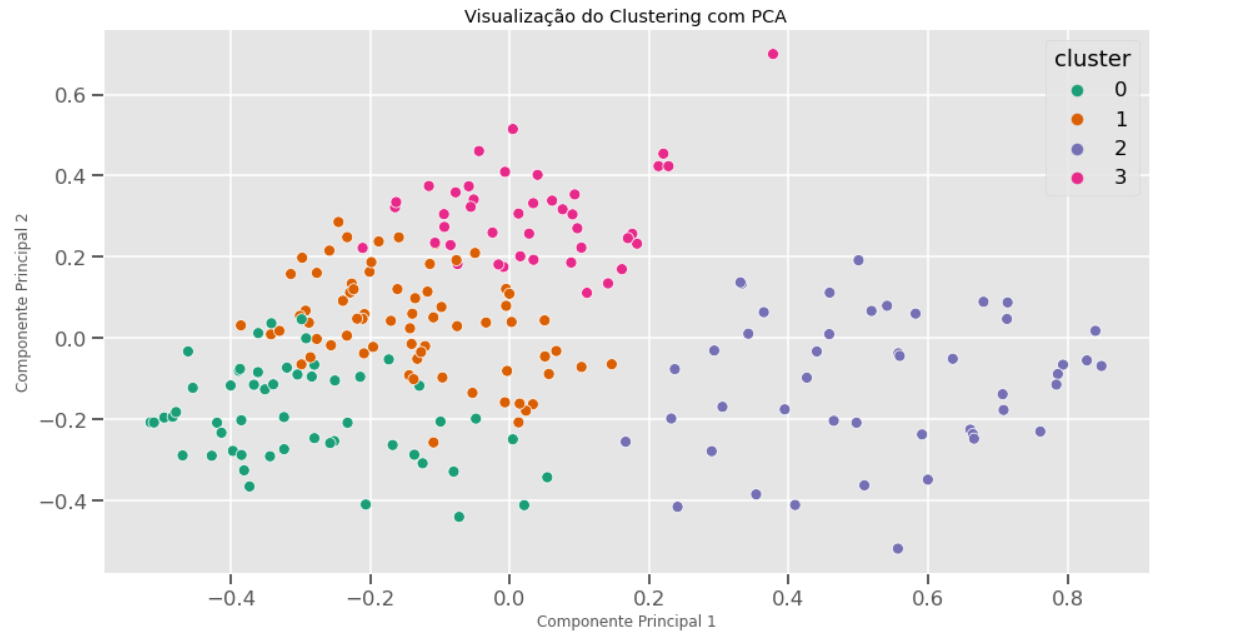


*Figura 4.4 – Dados clusterizados com o algoritmo t-SNE*

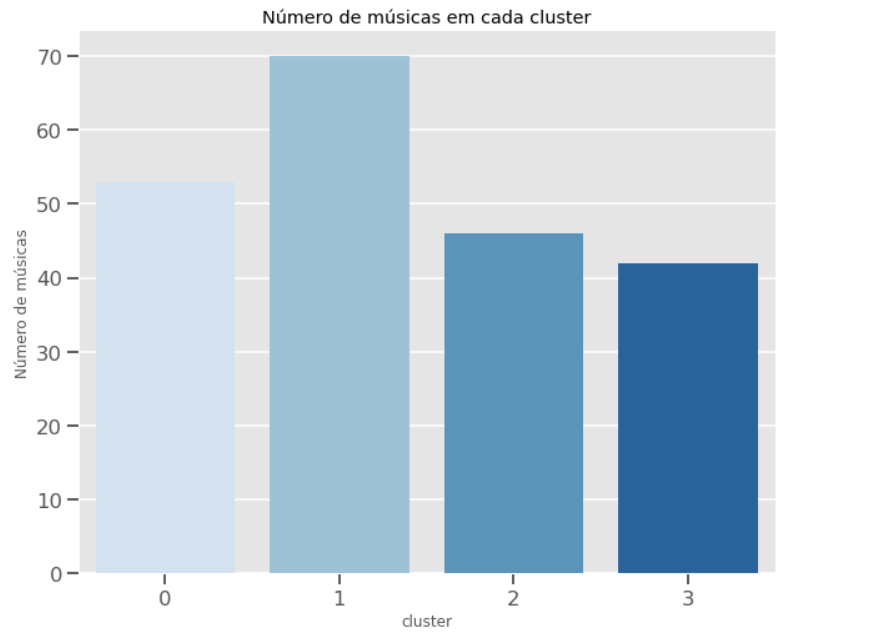
O PCA (Principal Component Analysis), apresentado abaixo, busca preservar grandes distâncias aos pares. (Figura 4.5 a 4.7)



*Figura 4.5 – Algoritmo PCA*



*Figura 4.6 – Dados clusterizados com o algoritmo PCA*



*Figura 4.7 – Músicas agrupadas em cada cluster de acordo com a similaridade das features*

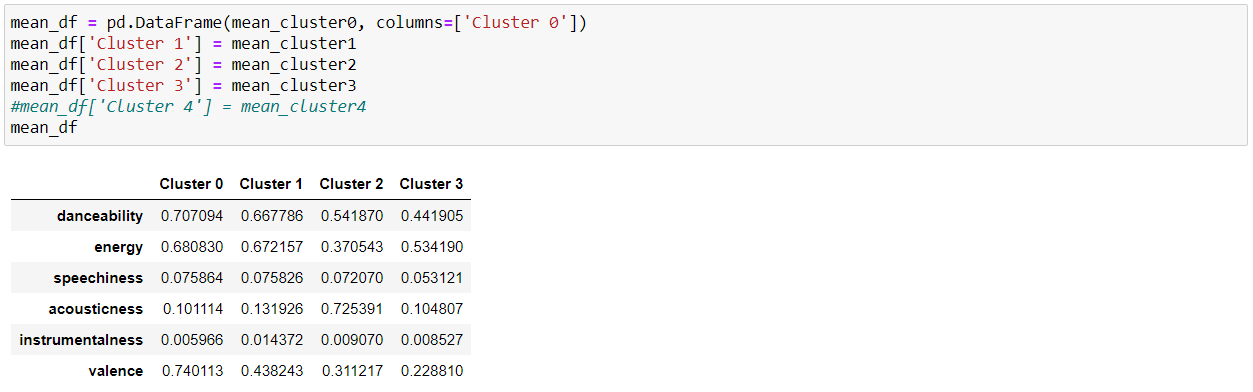
Percepções:

O “Cluster 0”, além de ser adequado para dançar, é o que conta com a maior quantidade de músicas agitadas e alegres. Muito provavelmente, as faixas de Rock/Pop predominam neste grupo.

O “Cluster 1” conta com músicas agitadas e estas nem sempre tendem a ser muito felizes. Apesar disso, é o grupo onde as faixas são um pouco adequadas para dançar.

O “Cluster 2” não é muito adequado para dançar, possui várias músicas acústicas e tem, predominantemente, faixas lentas. Este é o grupo onde estão presentes, em sua maioria, faixas mais tranquilas.

O “Cluster 3” possui músicas não adequadas para dançar, as músicas são predominantemente calmas e tristes. Este é o grupo onde as faixas tendem a ser mais tranquilas. (Figura 4.8)



*Figura 4.8 – Valores das features pré-definidas em cada cluster*

Para ilustrar mostramos abaixo umas das percepções citadas acima. Na figura abaixo estão as músicas presentes no Cluster 3, onde as faixas tendem a ser mais adequadas para dançar, as músicas são predominantemente barulhentas e felizes. Este é o grupo onde as faixas tendem a ser mais alegres e agitadas. (Figura 4.9)



*Figura 4.9 – Músicas presentes no cluster 3*

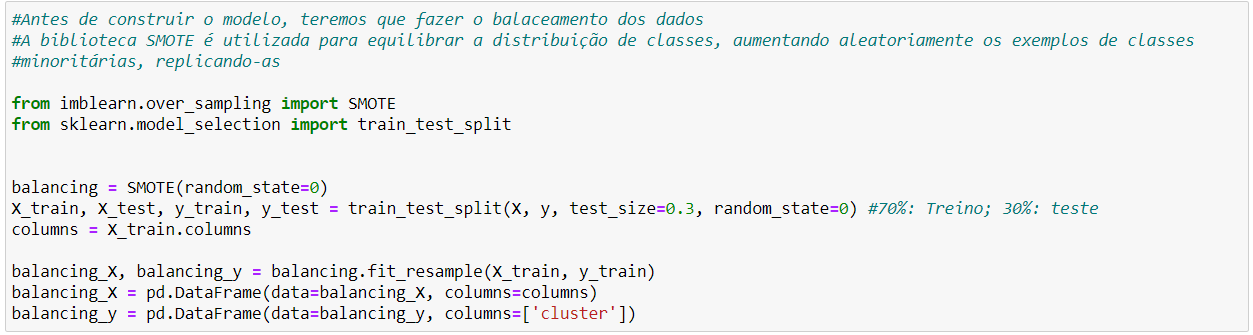
Após a etapa de clustering, o grupo de dados passou a ter uma classe, o que significa que precisamos utilizar algum modelo para prever um rótulo para cada uma das instâncias.

Pelo fato do número de músicas em cada cluster não ser equilibrado, o [SMOTE](https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html) foi usado para fazer um balanceamento nos dados de treino, ou seja, distribuí-los mais uniformemente.

A classificação desequilibrada envolve o desenvolvimento de modelos preditivos em conjuntos de dados de classificação que têm um desequilíbrio de classe severo.

O desafio de trabalhar com conjuntos de dados desequilibrados é que a maioria das técnicas de aprendizado de máquina irá ignorar e, por sua vez, terá baixo desempenho na classe da minoria, embora normalmente seja o desempenho na classe da minoria que é mais importante.

Uma abordagem para lidar com conjuntos de dados desequilibrados é sobreamostrar a classe minoritária. A abordagem mais simples envolve a duplicação de exemplos na classe minoritária, embora esses exemplos não adicionem nenhuma informação nova ao modelo. Em vez disso, novos exemplos podem ser sintetizados a partir dos exemplos existentes. Este é um tipo de [aumento](https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/) de [dados](https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/) para a classe minoritária e é referido como Técnica de Sobreamostragem Minoria Sintética , ou SMOTE, para abreviar. (Figura 4.10)



*Figura 4.10 – Balanceamento dos dados*

A validação cruzada foi usada para prever qual seria o modelo que melhor se ajustaria ao conjunto de dados. Foram testados os seguintes algoritmos: (Figura 4.11)

SVC (Support Vector Classification) com kernel linear

Random Forest Classifier

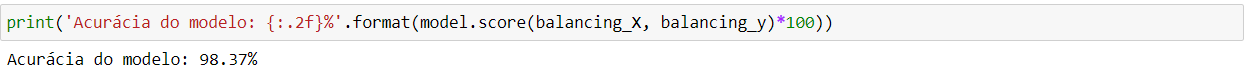
SVC (Support Vector Classification)

K-Neighbors Classifier



*Figura 4.11 – Seleção do Modelo*

A imagem acima nos mostra que, com uma média de 98.90% de acurácia, o algoritmo que mais se ajustou aos dados foi o SVC. Após usar o SVC para treinar o modelo, obtivemos uma acurácia de 98,37%.



Analisando os clusters, podemos ver que eles fazem sentido, e eu definitivamente consideraria essa uma boa combinação de músicas. Em sistemas de recomendação do tipo de produção, geralmente temos **dados do usuário** e outros recursos que podem tornar as recomendações mais sofisticadas e precisas.

# 6. Apresentação dos Resultados

Chega-se a mais importante fase do projeto, que é a análise e discussão dos resultados, onde será evidenciado os dados colhidos, e com esses dados serão feitas as recomendações musicais e pôr fim a criação das playlists.

Você já deve ter visto, por aí, alguém utilizando o termo "dar stream" em determinada música ou álbum, certo? O termo nada mais é do que "ouvir as músicas de determinado artista nas plataformas digitais". Atualmente é importantíssimo para a monetização de qualquer artista em tempos de música por streaming, mas somente se isso ocorrer em grande escala. Quanto mais vezes determinada música for tocada, mais dinheiro entra para a conta do artista, gravadora e os outros agentes responsáveis pela faixa. Imagine se tivéssemos o poder do conhecimento e criarmos algoritmos de recomendações de músicas e com essas recomendações o número de streaming triplicasse?

Com a plataforma de música digital temos milhares de opções de streaming musical. Extrair informações desses dados não é uma tarefa muito difícil: basta termos acesso a API Web e pronto. Com o conhecimento da plataforma e as ferramentas corretas conseguimos coletar os dados necessário para as recomendações musicais e a criação de playlists. (Figura 5).



*Figura 5 – Exemplos dos dados coletados no Spotify.*

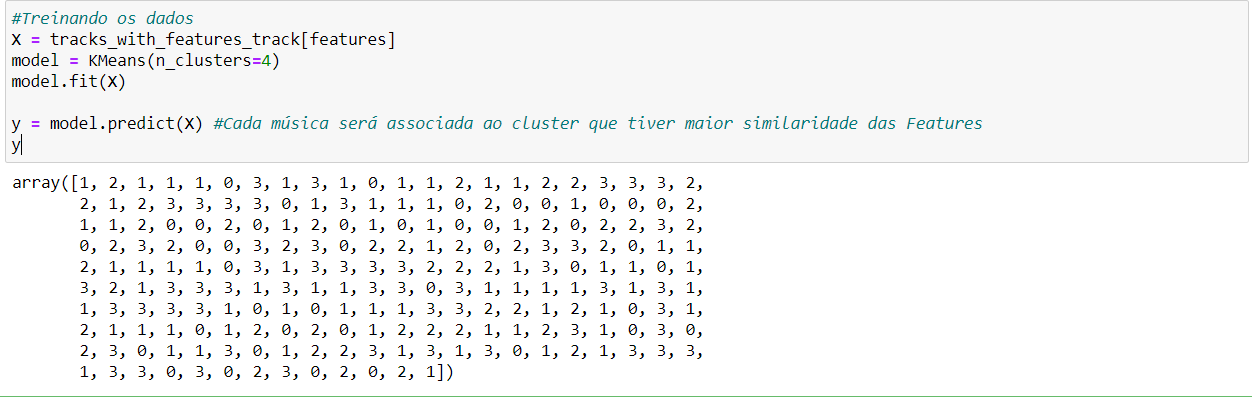
Um desafio que se põe, dessa forma, é encontrar meios de analisar esse tipo de dado e obter informações que possam direcionar para recomendações musicais com o gosto musical do usuário e até mesmo agregar novos gostos musicais para aquele usuário.

No intuito de contribuir com tal tarefa, foram acessados dados da própria plataforma Spotify, que consistem em milhões de músicas. O objetivo do projeto é, portanto, analisar uma coleção de *dados coletados, de um usuário, no spotify, e com esses dados conseguir realizar recomendações musicais e criar playlists baseadas nessas recomendações.*

Para tanto, os dados foram coletados da própria plataforma Spotify e, em seguida, agrupados em 4 clusters pelo modelo K-Means de aprendizado de máquina não supervisionado. Esses agrupamentos foram realizados baseado nos dados coletados da API do spotify. A API oferece a capacidade de extrair vários recursos de áudio de uma música. Os recursos utilizados para realizar o agrupamento das músicas no projeto foram as áudio features, elas já foram apresentadas no projeto. Escolhemos trabalhar com seis features, pois por padrão elas possuem os valores entre 0,0 e 1,0. Que são:

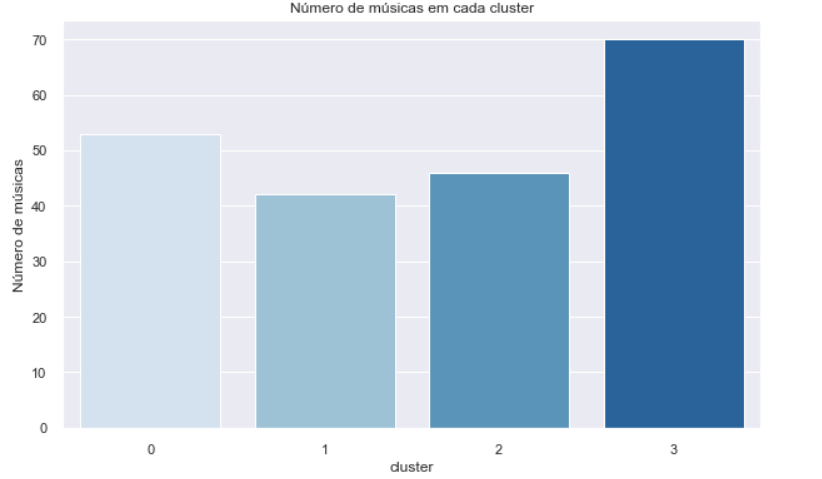
* danceability,
* energy,
* speechiness,
* acousticness,
* instrumentalness e
* valence.

O método utilizado consiste no agrupamento das músicas, onde ela será associada ao cluster que tiver maior similaridade das Features. (Figura 5.1)

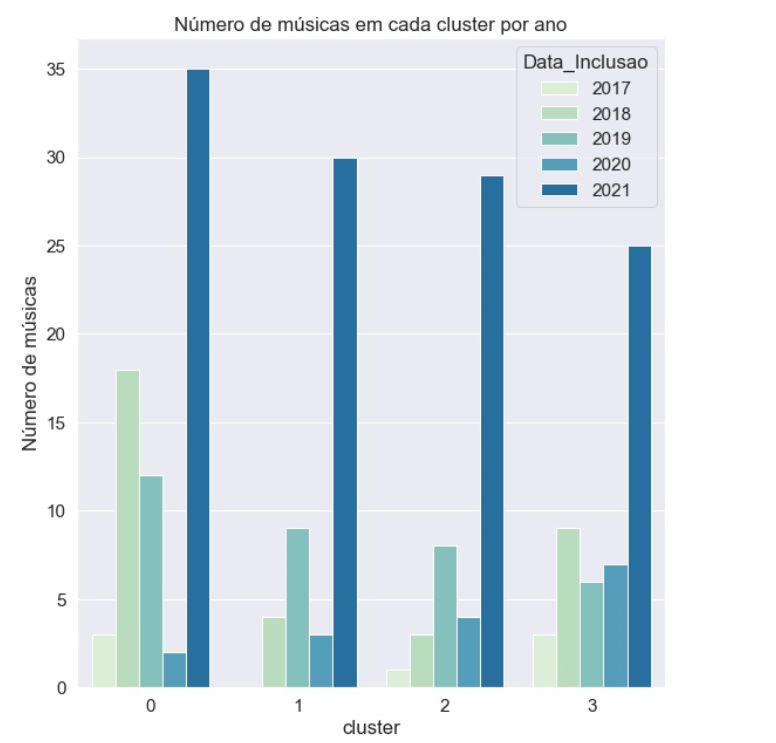


*Figura 5.1 - Associando as músicas aos cluster com similaridade das Features*

Conforme se verifica na Figura 5.2, o grupo com maior número de *músicas* foi o cluster 3. Um aspecto importante consiste na pouca diferença entre a quantidade de *músicas* agrupadas no cluster 0, no cluster 1 e no cluster 2.

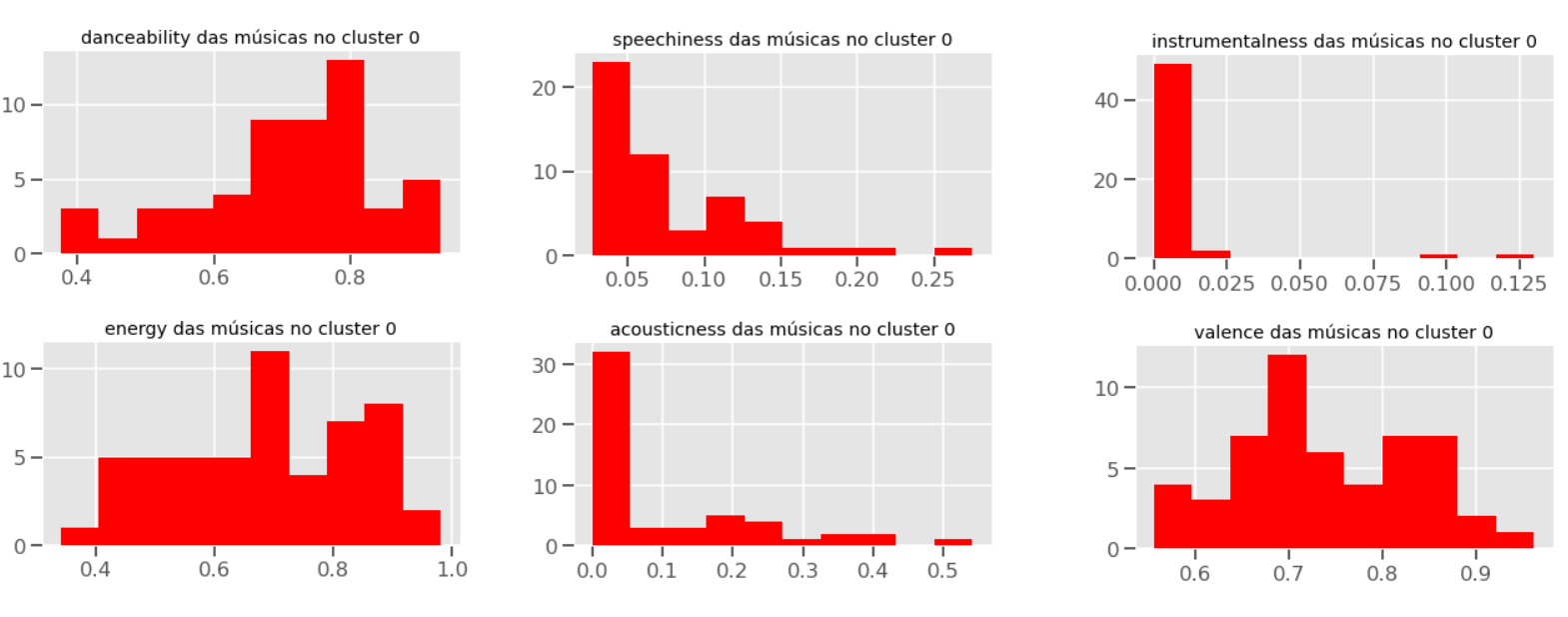


*Figura 5.2 – Quantidade de músicas por cluster (KMeans).*



*Figura 5.3 – Quantidade de músicas por cluster e por ano (KMeans).*

No tocante aos gráficos de quantidades, iremos considerar à distribuição das músicas. Da análise do gráfico, verifica-se que as músicas agrupadas no cluster 0 são mais agitadas, cabendo destacar que, no que tange ao cluster 3, praticamente temos o agrupamento de músicas mais calmas, menos alegres, já que a feature valence tem o valor menor que 0,3. (Figura 5.4 e 5.5)



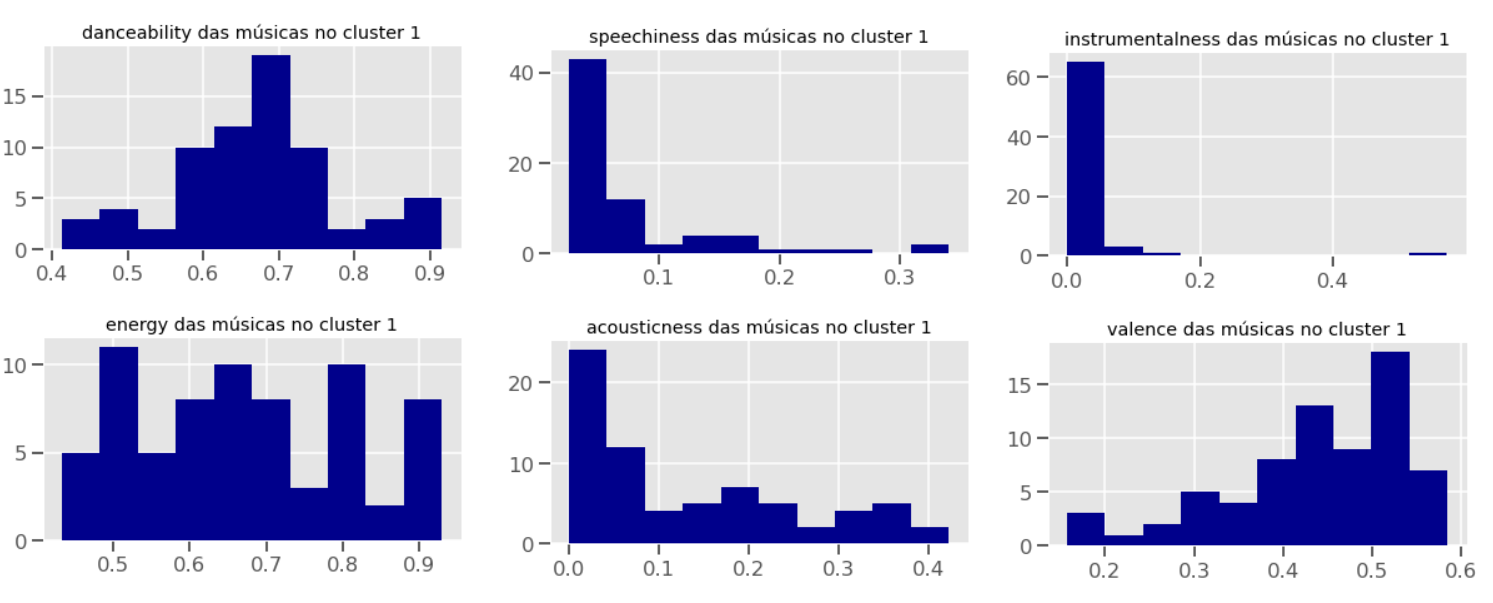
*Figura 5.4 – Valor das features no cluster 0 (KMeans).*

As músicas agrupadas no cluster 0 estão apresentadas na Figura 5.5.



*Figura 5.5 – Músicas agrupadas no cluster 0 de acordo com a similaridade das features.*

Continuando a análise, temos os gráficos relativos ao *cluster 1* onde temos músicas menos alegres, porém com um nível considerável de danceability e energy, portanto temos um mix de músicas calmas e agitadas, ou seja, algumas boas para dança e outra nem tanto (Figuras 5.6 e 5.7).



*Figura 5.6 – Valor das features no cluster 1 (KMeans).*

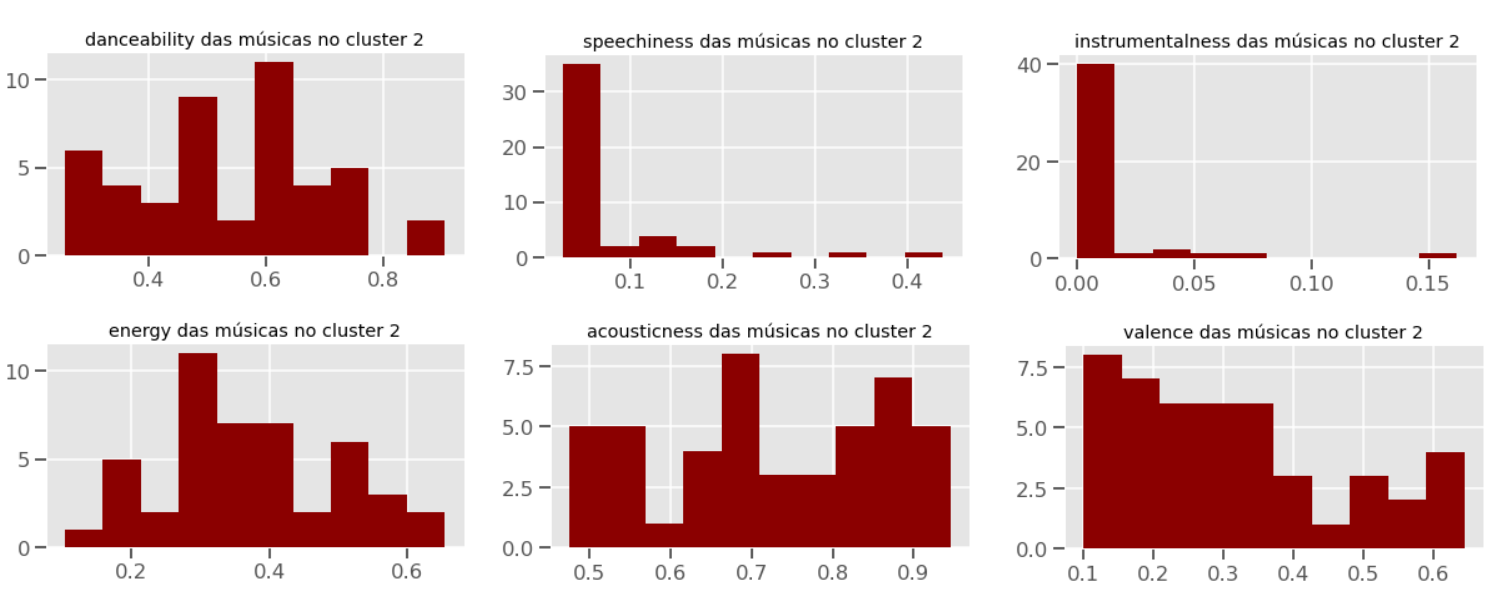
O gráfico de figura 5.6 evidencia as features no cluster 1. Observa-se que a feature energy possue valores distribuídos, já na feature danceability os valores ficaram mais centralizados e a valence possui valores abaixo de 0,6, portanto digamos que as músicas presentes no cluster 1 são variadas, ou seja, possuem músicas boa para dançar, mas também possue músicas não tão alegres.

As músicas agrupadas no cluster 1 estão apresentadas na Figura 5.7.



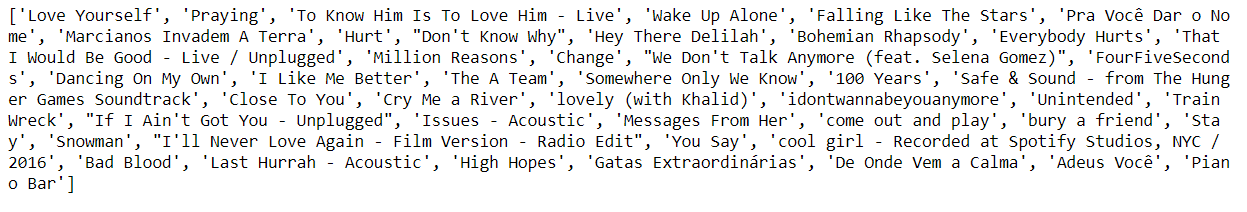
*Figura 5.7 – Músicas agrupadas no cluster 1 de acordo com a similaridade das features.*

Temos os gráficos relativos ao *cluster 2* com músicas tristes e românticas, um nível baixo de valence e energy, porém verificamos um nível de dancability com valores equilibrados entre 0,2 e 0,6 o que pode ocasionar em músicas pouco adequadas para dançar. Já um ponto interessante é de que as músicas possuem valores altos para a feature acousticness (Figuras 5.8 e 5.9).



*Figura 5.8 – Valor das features no cluster 2 (KMeans).*

As músicas agrupadas no cluster 2 estão apresentadas na Figura 5.9.



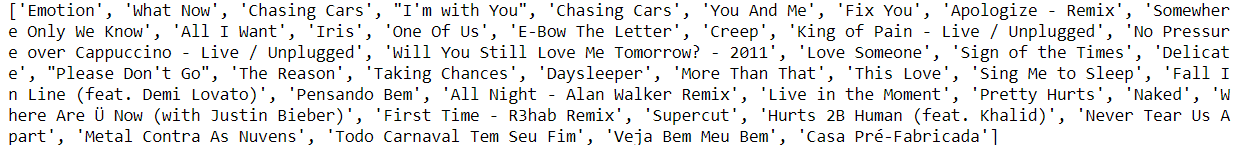
*Figura 5.9 – Músicas agrupadas no cluster 2 de acordo com a similaridade das features.*

O cluster 3, último grupo em quantidade de *músicas*, apresentou valores baixos para a feature valence e energy, onde temos uma quantidade de músicas concentradas entre os valores 0,1 e 0,4, portanto temos músicas tristes, músicas mais depressivas. (Figuras 5.10 e 5.11)



*Figura 5.10 – Valor das features no cluster 3 (KMeans).*

As músicas agrupadas no cluster 3 estão apresentadas na Figura 5.11.



*Figura 5.11 – Músicas agrupadas no cluster 3 de acordo com a similaridade das features*

Sob outra perspectiva, as de nuvens de palavras proporcionam uma visualização mais direta dos termos considerados de maior relevância. Para isso, foram elaboradas as seguintes nuvens de *palavras*: a) nuvem gerada a partir dos nomes das músicas agrupadas no cluster 0 (Figura 5.12); b) nuvem gerada a partir dos nomes das músicas agrupadas no cluster 1 (Figura 5.13); c) nuvem gerada a partir dos nomes das músicas agrupadas no cluster 2(Figura 5.14); e d) nuvem gerada a partir dos nomes das músicas agrupadas no cluster 3 (Figura 5.15)



*Figura 5.12 – Nuvem de palavras das músicas agrupadas no cluster 0.*

Alguns dos termos mais frequentes (de maior tamanho) apresentados na nuvem de *palavras* relativa as músicas agrupadas no cluster 0 já eram esperados por estarem ligados às músicas mais alegres, mais dançantes, tais como: *Remix, Beat, Fell.*

Já para os termos apresentados na nuvem de palavras relativas as músicas agrupadas no cluster 1 temos: Love, Cry, Fever e Best. Uma curiosidade é ver a palavra feat se repetindo tanto no cluster 0 quanto no cluster 1 o que nos dá uma ideia de gosto musical de parceria entre os artistas.



*Figura 5.13 – Nuvem de palavras das músicas agrupadas no cluster 1*

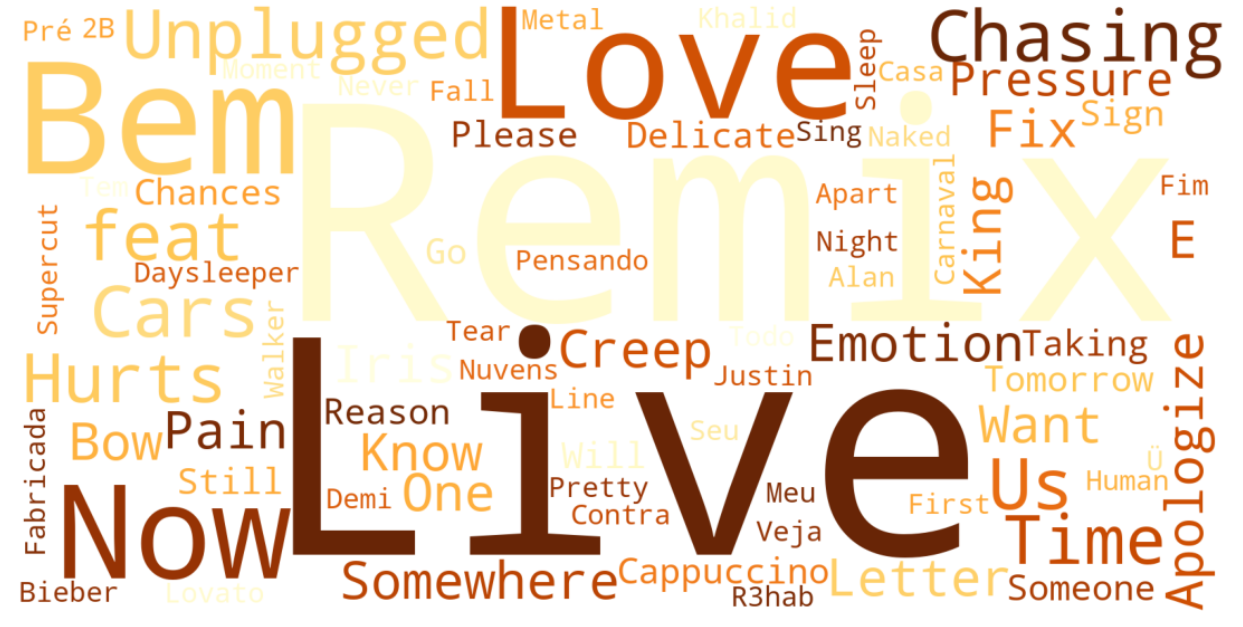
Agora vejamos os termos apresentados na nuvem de palavras relativas as músicas agrupadas no cluster 2, os termos mais frequentes são: Acoustic, Know, Love, Hurt, Você, Live. Com esses termos podemos ver que as músicas pertencentes ao cluster 2 são mais tristes e que dão uma ideia de sofrimento.



*Figura 5.14 – Nuvem de palavras das músicas agrupadas no cluster 2*

Verificamos que alguns termos se repetiram, no caso tivemos o termo Love aparecendo no cluster 0, no cluster 1 e no cluster 2 o que nos dá uma idéia de músicas mais romantizadas.

E por último temos os termos apresentados na nuvem de palavras relativas as músicas agrupadas no cluster 3 que são: Live, Now, Remix, Hurts



*Figura 5.15 – Nuvem de palavras das músicas agrupadas no cluster 3*

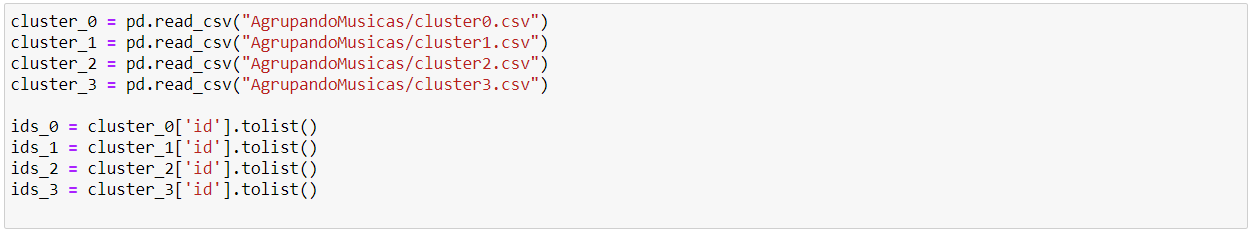
E novamente tivemos o termo Love aparecendo o que reforça a idéia de músicas mais romantizadas.

Dado a realização de agrupamentos musicais onde temos a similaridade dos valores nos clusters criados iremos realizar a o processo de recomendações de músicas e pôr fim a criação das playlists. Para essa etapa foram criados os arquivos cluster\_0, cluster\_1, cluster\_2 e cluster\_3 no formato csv. Dentro dos arquivos existem as músicas agrupadas pelo modelo kmeans, lembrando que elas pertencem as playlists do usuário ‘betatunes’.(Figura 5.16)



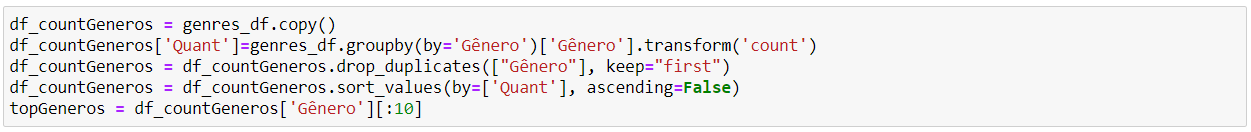
*Figura 5.16 – Arquivos criados com as músicas agrupadas*

Para o processo de recomendações musicais utilizamos o método sp.recommendations(seed\_tracks=[i],seed\_genre=topGeneros,limit=100) pertencente a API Spotify, como podemos ver o método exige dois parâmetros que são os track\_ids e o gênero. O primeiro parâmetro retiramos dos arquivos informados anteriormente. (Figura 5.17)



*Figura 5.17 – A forma como foi coletado o primeiro parâmetro que será utilizado para as recomendações musicais*

Já para o segundo parâmetro buscamos os top 10 gêneros musicais presentes nas playlists do usuário. (Figura 5.18)



*Figura 5.18 – Foi criado um DataFrame com os gêneros musicais agrupando-os pela quantidade*

Com todos os parâmetros devidamente separados o próximo passo para as recomendações musicais foi a criação da função \_get\_track\_Rec\_df(sp, track\_ids, topGeneros) (Figura 5.19)



*Figura 5.19 – Função criada para buscar as músicas recomendadas. O resultado retorna um DataFrame com as recomendações*

Agora que estamos com as recomendações musicais em um DataFrame, chegamos na parte final do projeto que no caso será a criação das playlists. A quantidade de playlists a serem criadas está relacionada com a quantidade de clusters recomendados pelo método Elbow, que no caso pelo gráfico vimos que seriam 4. Para a criação da playlist foi criada a função \_get\_criaPlaylist\_df(sp, username,df\_search).

(Figura 5.20)



*Figura 5.20 – Função para a criação das playlists*

Para a criação das playlists passamos o parametro sp, username e o DataFrame com as músicas recomendadas utilizando os Ids das músicas de cada cluster. (Figura 5.21)



*Figura 5.21 – Mostrando a forma como foram chamadas a funções de recomendação e criação de playlists*

Por fim segue abaixo a figura na qual mostramos as playlists criadas no Spotify.



É importante registrar que esses resultados ainda podem ser aprimorados pela otimização na escolha do número k de clusters utilizado no modelo K-Means de agrupamento, bem como pelo aperfeiçoamento da função de pré-processamento das músicas mediante a inclusão de tratamento de outras features existentes no spotify, assim como a utilização de Inteligência Artificial.

Por fim, como o presente projeto tratou de algoritmo de aprendizado de máquina não supervisionado, em que a interpretação dos resultados é dificultada pela falta de transparência nos critérios de agrupamento utilizados pelo algoritmo, é relevante lembrar que a validação de seus resultados por especialistas da área é de fundamental importância.

# 7. Links

Vídeo de apresentação do projeto:

Scripts criados em Python:

Repositório dos dados utilizados no projeto:

# REFERÊNCIAS

1. PYTHON. Disponível em: <[https://www.python.org/>](https://www.python.org/).
2. SPOTIFY DEVELOPER. Disponível em: <<https://developer.spotify.com/>.> Acessado em: 15/02/2021
3. JUPYTER. Disponível em: <[https://jupyter.org/>](https://jupyter.org/).
4. SEABORN. Disponível em: <[https://seaborn.pydata.org/>](https://seaborn.pydata.org/).
5. SCIKIT LEARN. *Sklearn.decomposition.TruncatedSVD*. Disponível em: <<https://scikit>[-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html) > . Acesso em 10/03/2021.
6. SANTANA, Felipe. *Algoritmo K-means: Aprenda essa Técnica Essencial através de Exemplos Passo a Passo com Python.* Disponível em:

<https://minerandodados.com.br/algoritmo-k-means-python-passo-passo/>. Acesso em 04/03/2020.

1. WIKIPEDIA. *Silhueta (agrupamento).* Disponível em:

<https://en.wikipedia.org/wiki/Silhouette\_(clustering)>. Acesso em: 11/03/2021.

[8] PLOTLY PYTHON. Disponível em: <<https://plotly.com/python/>>. Acesso em: 02/03/2021

[9] towardsdatascience – Unsupervised learning Disponivel em: <<https://towardsdatascience.com/unsupervised-learning-with-scikit-learn-spotify-api-and-tableau-public-50fcecf3bdf5>.>. Acesso em: 03/03/2021