Insper

Insper: Instituto de Ensino e Pesquisa

Ellen Shen
Enzo Neto
Gabriel Huerta
Giulia Castro

CIÊNCIA DOS DADOS
Projeto 3: Análise de dados do Spotify

São Paulo
2019

Introdução:

As três técnicas que irão ser introduzidas se baseiam em uma base de dados do **spotify**, enquanto uma delas foi puxada utilizando o API do próprio spotify, outra foi encontrada do site “kaggle” (playlists de músicas top_2017 e top_2018).

Para cada técnica fizemos uma pergunta diferente.

- *Clusterização:* "Dado uma playlist com gêneros musicais diferentes. Um usuário está escutando uma determinada música desta playlist, qual seria a melhor música para ser recomendada como a próxima?"
- *Decision Tree:* "Dado uma playlist qualquer compartilhada no spotify, com 3 usuários contribuintes. É possível atribuir uma música nova para cada um dos usuários de acordo com seu gosto musical?"
- *Naive-Bayes:* "Analisando uma base que contém músicas pré-classificadas, é possível classificar com precisão novas músicas não pertencentes da mesma base?"

Clusterização:

Para a técnica de clusterização inicialmente foram importadas as seguintes bibliotecas: matplotlib.pyplot, pandas, numpy, mpl_toolkits.mplot3d, seaborn e sklearn. Logo após foram importadas as bases de dados e divididas entre a base que seria utilizada para o treinamento – chamada de top_2017_2018_treinamento – e a base que seria utilizada para o teste do código criado – chamada de top_2017_2018_teste.

Depois desse processo inicial, começamos uma análise exploratória básica dos dados. Desse modo, foi plotado um scatterplot com as três variáveis escolhidas para o nosso estudo, como mencionado na introdução, para a visualização da distribuição das músicas que compunham a base de dados no espaço.

Através do comando *cluster.MiniBatchKMeans* a equipe definiu por trabalhar com 5 grupos distintos, e logo após foi realizado um *fit* passando como argumento as colunas das variáveis que foram definidas anteriormente (danceability, energy e valence). Utilizando o comando *.predict*, o grupo obteve a separação das músicas nos diferentes grupos pré-estabelecidos. O resultado foi guardado na variável *saida_2017_2018* e copiados para construir um novo dataset no qual possuía a coluna de saída com os seus respectivos clusters, o qual foi chamado de: *musicas_2017_2018*.

Após esse processo foi plotado um outro gráfico de dispersão que separava as músicas pertencentes a cada grupo por cores diferentes. Ou seja, nesse novo gráfico obtivemos a mesma distribuição das músicas no espaço, porém com pontos de cores diversas onde o Grupo 1 foi representado pela cor vermelha, o Grupo 2 pela cor verde, o Grupo 3 pelo azul, Grupo 4 pelo preto e o Grupo 5 pelo roxo. Chegando ao resultado representado na Figura 1 abaixo.

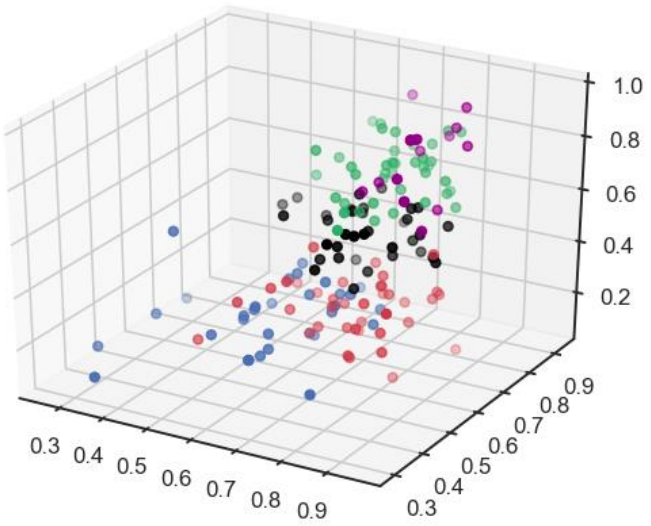

Figura 1 – Gráfico de dispersão das músicas separadas nos grupos

Apesar do gráfico possibilitar uma ótima observação da distribuição dos Grupos, ele não possibilita uma precisão nos valores utilizados para a construção da divisão. Tendo isso em vista, foi realizado um Pairplot para cada grupo para entendermos a lógica utilizada pelo computador para fazer a divisão. Sendo assim foi concluído que:

- Grupo 1: Apresenta dançabilidade alta, porém a valência e a energia médias. Portanto, são músicas dançantes, mas sem muito barulho e não são nem muito alegres nem tristes.

- Grupo 2: Possui as três variáveis com valores altos. Conclui-se que são músicas dançantes, barulhentas e alegres.

- Grupo 3: Pode-se observar que esse grupo apresenta as músicas com valência muito baixa. Assim, conclui-se que as músicas desse grupo são negativas e mais deprimentes.

- Grupo 4: Como pode ser visto as músicas pertencentes a esse grupo apresentam dançabilidade e energia altas e valores de valência medianos. Conclui-se, então, que as músicas são dançáveis, bem rápidas e barulhentas, porém não são tão positivas.

- Grupo 5: Nota-se pelo gráfico de dispersão que essas músicas apresentam valência, energia e dançabilidade altos. Ou seja, as músicas desse grupo são no geral bem dançantes, bem energéticas e muito positivas.

Com a conclusão dos resultados obtidos, começamos a trabalhar com a base de dados do teste, repetindo os mesmos processos feitos n treinamento. Ou seja, inicialmente plotamos um gráfico de dispersão das músicas para a visualização dos dados; depois definimos os 5 grupos com os mesmos comandos utilizados anteriormente; e em seguida foi plotado o gráfico final de dispersão com a mesma divisão de cores. Chegando ao resultado representado na Figura 2.

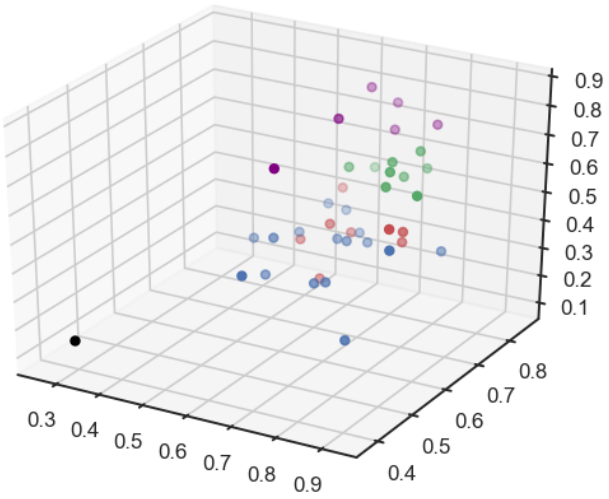

Figura 2 - Gráfico de dispersão das músicas de teste separadas nos grupos

Basta uma análise rápida do gráfico para perceber que os grupos definidos pelo código não correspondem com os Grupos definidos anteriormente; o Grupo 4 (preto), por exemplo, era um Grupo com músicas dançáveis, rápidas e barulhentas, apesar de não serem tão positivas, porém nessa nova distribuição esse mesmo Grupo contém apenas 1 música que é negativa, não dançável e bem pouco energética.

Conclui-se que para essa técnica ser viável seria necessário possuir um gabarito.

Decision Tree

Para implementar esta técnica foram utilizadas as bibliotecas do python pandas e sklearn (incluindo as sub-bibliotecas preprocessing e model_selection). O primeiro passo para fazer a árvore foi coletar a base de dados do spotify: através da conta de desenvolvedor foram obtidas as características de áudio de 100 músicas pertencentes aos top charts de 2018 e 50 outras do “top songs” de outubro de 2019 do próprio aplicativo. É importante apontar que, por motivos práticos, o csv foi manipulado manualmente para retirar algumas variáveis que se julgaram pouco significativas e juntar as duas bases de dados.

A seguir, para poder existir uma variável de input, as 150 músicas foram classificadas em três categorias cada uma representando a preferência de um membro do grupo: músicas de que Enzo gostava recebiam 0; Ellen, 1 e Gabriel, 2.

O programa consistiria, então, em um ‘split’ de treino e teste com um ‘test size’ pequeno para conseguir mais casos para treino; e, em seguida o código principal do classificador da árvore e seu ‘fit’. No entanto, ao checar o score do teste, obteve-se um valor baixo; próximo de 30%.

Por acaso, ocorreu um erro no código e ao tentar consertá-lo foi retirada a variável “instrumentalness”, que mede o quão instrumental é a música. Sem essa variável o teste chegou a obter um acerto de 53%.

Foi possível concluir que a variável estava afetando significativamente a performance do classificador e pôde-se entender a importância de escolher bem as variáveis utilizadas.

Adicionalmente, como um teste extra, foram pegas três músicas; diretamente das playlists de cada um dos três membros da classificação, e suas características foram testadas com a função ‘predict’ da decision tree e, para a surpresa e alegria do grupo todas as três músicas foram classificadas corretamente como sendo de seus respectivos “donos”.

Em conclusão, a técnica da Decision Tree apresentou resultados decentes, mostrando-se viável. Ainda mais se, em futuras iterações fossem estudadas as diferentes variáveis e suas influências mais minuciosamente.

Naive Bayes

Para a técnica de naive bayes inicialmente foram importadas as seguintes bibliotecas: matplotlib.pyplot, pandas, numpy e sklearn. Logo após foram importadas a base de dados e dividida entre duas variáveis “X” e “Y”, que irão servir como parâmetros mais tarde. A variavel “X” é basicamente um loc das três colunas contendo os dados que serão analisados, já a variável “Y” é a coluna de saída (contida no dataframe) com as classificações anteriormente feitas pelo método de clusterização.

Após separar as colunas em suas respectivas variáveis, dividimos a parte que seria utilizada para o treinamento – chamada de x_treinamento – e a parte que seria utilizada para o teste do código criado – chamada de x_teste (o mesmo foi feito para a variável “Y”).

Utilizando o comando MultinomialNB passamos os parâmetros “X” e “Y”, realizando um “.fit” (treinando o código), e então fizemos um “.predict” com a base de teste, resultando assim em uma classificação feita pelo código.

Para entender melhor como a classificação foi feita, foi feito um “value_counts” da coluna “saida” da base de treinamento, para verificar a proporção contida em cada grupo, percebendo que os grupos com a maior contagem eram mais suscetiveis a serem classificados na base de testes.

```
y_treinamento.saida.value_counts()
```

```
2    32
0    32
3    28
1    21
4    15
Name: saida, dtype: int64
```

Figura 3 – Proporção dos dados de treinamento.

Dentro da biblioteca *sklearn*, utilizamos os comandos que calculam a acurácia e precisão. Obtivemos uma acurácia de 46% e uma precisão de 23%. Chegando à conclusão de que esta tecnica não é confiavel pois tem um baixo indice de acerto. Para uma futura iteração, faria sentido usar mais variáveis para serem comparadas entre si no classificador, dado que ao tentar diminuir as colunas escolhidas, foi obtido uma porcentagem ainda menor de precisão. Outra alternativa seria utilizar uma base maior de dados.

Divisão do projeto:

- *Clusterização*: Ellen, Giulia e Enzo
- *Decision Tree:* Gabriel
- Naive-Bayes: Ellen e Enzo
- Relatório: Todos