Projeto Final - Uma análise sobre o métodos de normalização de dados e seus impactos em um sistema de recomendação de músicas

Nomes:

Davi Mattos - 119133049

Matéria: ICP248 - Computação Científica e Análise da Dados

Profª: Joao Antonio Recio Da Paixao

Sumário

1. Introdução

O projeto tem como objetivo mostrar o impacto dos métodos de normalização de dados em três frentes, visualização (PCA), como os dados se comportam se tentarmos exibi-los em 2D e 3D, classificação das músicas (Clusterização), será que para cada método será feita uma classificação diferente, e por último recomendação (Clusterização), para uma mesma música, será que as recomendações serão muito diferentes, ou recomendam as mesmas músicas.

2. Dataset

O dataset escolhido foi o 🎹 Spotify Tracks Dataset. Um dataset de músicas do Spotify, que contém dados da música (id, nome, intérpretes, álbum) e recursos de áudio (popularidade, duração, volume, etc) para cada música.

3. Implementação

Para realização do trabalho foi utilizado o jupyter notebook, juntamente com a linguagem de programação Python, e por fim a utilização das bibliotecas Pandas, SciKit Learn, Math e Matplotlib.

Dos tópicos abordados em sala de aula e utilizados no projeto:

Análise de Componentes Principais (PCA)

A Análise de Componentes Principais (PCA) é uma técnica estatística poderosa utilizada para simplificar dados complexos. Em nosso projeto, que lida com diversas características musicais (como *danceability*, *energy*, *loudness*, etc.), o PCA nos permite reduzir a dimensionalidade desses dados, ou seja, condensar muitas variáveis em apenas duas ou três (os "componentes principais"), mantendo o máximo de informação original possível.

O principal objetivo de aplicar o PCA neste trabalho foi a visualização dos agrupamentos de músicas (clusters) em gráficos 2D e 3D, tornando possível enxergar visualmente a separação dos grupos formados pelo K-Means.

O processo do PCA pode ser dividido em quatro etapas fundamentais:

1. **Centralização dos dados:**
   O primeiro passo é ajustar os dados para que todas as variáveis tenham uma média igual a zero. Isso é feito subtraindo a média de cada coluna de seus respectivos valores. Essa etapa garante que o PCA foque apenas na variação dos dados e não em suas médiasl:

$$\overline{X} = X - \mu$$

2. **Cálculo da matriz de covariância:**

   Em seguida, calculamos a matriz de covariância para entender como as diferentes características musicais se relacionam entre si. Por exemplo, será que músicas com alta "energia" também tendem a ter alto "volume"? A matriz de covariância captura essas relações

$$\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$$

3. **Cálculo dos autovalores e autovetores:**
   A partir da matriz de covariância, extraímos seus autovetores e autovalores.

   Autovetores: Representam as direções dos novos eixos (os componentes principais) no espaço de dados. Cada autovetor aponta na direção de máxima variância dos dados, de forma que eles são ortogonais entre si.

   Autovalores: Indicam a quantidade de variância que cada autovetor (componente principal) captura. Autovalores maiores correspondem aos componentes mais importantes

$$\Sigma v = \lambda v$$

   Onde (λ) representa a variância por cada componente, e (v) o vetor diretor da nova base.

4. **Projeção dos dados:**
   Por fim, projetamos os dados originais nos novos eixos definidos pelos autovetores associados aos maiores autovalores. Se quisermos visualizar os dados em 2D, por exemplo, selecionamos os dois autovetores com os maiores autovalores e transformamos nossos dados para esse novo sistema de coordenadas

$$X_{\text{PCA}} = \tilde{X}V_k$$

   Onde (V_k) contém os (k) autovetores associados aos maiores autovalores.

Ao aplicar o PCA, conseguimos uma representação simplificada e visual dos clusters de músicas.

Método do Cotovelo (Elbow Method)

Para encontrar o número ideal de clusters (k), usamos o Método do Cotovelo, que avalia a **Soma dos Erros Quadrados Internos**:

$$WCSS = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

Onde o valor ideal de (k) é o ponto onde o acréscimo de novos grupos gera pequenas alterações no erro — formando o "cotovelo" no gráfico.

No projeto, implementamos uma função que calcula a distância de cada ponto do gráfico WCSS até a reta entre os extremos (1 e 20 clusters):

$$\text{dist} = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}$$

Essa é a distância perpendicular entre um ponto (x_0, y_0) e a linha reta $(x_1, y_1) \to (x_2, y_2)$, e o ponto com maior distância indica o melhor valor de (k).

E dos tópicos não abordados em sala de aula mas foram utilizados no projeto:

Algoritmo de Agrupamento K-Means

O K-Means é um algoritmo de aprendizado não supervisionado que divide os dados em (k) grupos, baseando-se na minimização da variância interna dos grupos.

Seu funcionamento ocorre em três etapas principais:

1. **Inicialização:** Seleção aleatória de (k) centróides.
2. **Atribuição:** Cada ponto é atribuído ao cluster com centróide mais próximo:
$$\text{distância} = \|x_i - \mu_j\|_2$$
3. **Atualização:** Recalcula-se o centróide como a média dos pontos atribuídos:
$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

O processo se repete até que os centróides se estabilizam (convergem). No projeto, o K-Means é implementado utilizando a biblioteca SciKit Learn

```
KMeans(n_clusters=ncluster1, random_state=14)
```

Pré-processamento dos Dados

Antes de aplicar técnicas de redução de dimensionalidade (PCA) e clusterização (KMeans), foi necessário padronizar os dados numéricos. O pré-processamento é essencial para evitar que variáveis com escalas diferentes (como duração em milissegundos e valência de 0 a 1) influenciam de maneira desproporcional os resultados. Neste trabalho, foram aplicadas quatro abordagens distintas:

Robust Scale

Utiliza a mediana e o intervalo interquartílico (IQR) para centralizar e escalar os dados:

$$X_{\text{robusto}} = \frac{X - \text{mediana}(X)}{\text{IQR}(X)}$$

Esse método é particularmente útil para lidar com **outliers**, já que a mediana é menos sensível a valores extremos.

Min-Max Scale

Transforma os dados para o intervalo [0,1] usando:

$$X_{\text{minmax}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

É útil quando desejamos que todos os atributos contribuam igualmente, mantendo a **forma da distribuição original**. No entanto, é sensível a outliers, pois valores extremos afetam o intervalo total.

Standard Scale

Padroniza os dados subtraindo a média e dividindo pelo desvio padrão:

$$X_{\text{padronizado}} = \frac{X - \mu}{\sigma}$$

Esse processo gera uma distribuição com média 0 e desvio padrão 1. É uma escolha comum para algoritmos que assumem normalidade ou que usam distâncias euclidianas.

MaxAbs Scale

Divide cada valor pelo valor absoluto máximo da variável:

$$X_{\text{maxabs}} = \frac{X}{|X_{\max}|}$$

Preserva a **distribuição dos sinais (positivo/negativo)** e não centraliza os dados. É útil para algoritmos sensíveis à magnitude, mas não ao deslocamento.

Análises

Para uma visualização mais completa sobre o dataset, o notebook utilizado pode ser encontrado em GitHub - Trabalho Final CoCAdA ou no seguinte site

Recomendações de Músicas

Para ver o resultado da recomendação, basta entrar neste colab, aqui tem recomendações para a música *"Chop Suey! - System Of A Down"*

Para implementar o sistema de recomendação de músicas, utilizamos uma abordagem baseada em agrupamento (clusterização) e comparação de similaridade entre vetores.

Inicialmente, os dados numéricos das músicas foram normalizados com diferentes técnicas e, posteriormente, agrupados com o algoritmo K-Means, que separou faixas com características semelhantes em clusters.

A partir da música escolhida pelo usuário, identificamos o cluster ao qual ela pertence e comparamos suas características numéricas (como valence, danceability, acousticness etc.) com as das outras músicas do mesmo cluster. Para isso, utilizamos a similaridade do cosseno, que mede o ângulo entre dois vetores no espaço, refletindo o quanto os perfis das músicas se alinham em termos de padrão, independentemente de suas magnitudes absolutas. Isso se mostrou mais eficaz do que a distância euclidiana, pois a recomendação busca identificar músicas com comportamentos semelhantes, mesmo que em intensidades diferentes. Dessa forma, são recomendadas músicas com características próximas à escolhida, respeitando a estrutura de agrupamento revelada pela clusterização.

Exemplo: Dado dois vetores $\vec{a}, \vec{b} \in \mathbb{R}^n$, onde

$$\vec{a} = [1, 2, 3] \qquad \vec{b} = [10, 20, 30]$$

Claramente, $\vec{b} = 10 \cdot \vec{a}$: mesma direção, magnitude 10x maior.

- **Distância Euclidiana**:

$$d(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

$$= \sqrt{(10-1)^2 + (20-2)^2 + (30-3)^2}$$

$$= \sqrt{81 + 324 + 729}$$

$$= \sqrt{1134} \approx 33.63$$

- **Similaridade do Cosseno**:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum a_i^2} \cdot \sum b_i^2}$$

$$= \cos(\theta) = \frac{1 \cdot 10 + 2 \cdot 20 + 3 \cdot 30}{\sqrt{1^2+2^2+3^2} \cdot \sqrt{10^2+20^2+30^2}}$$

$$= \frac{140}{\sqrt{14} \cdot \sqrt{1400}} = 1$$

A similaridade do cosseno preserva relações proporcionais entre as características musicais, o que é ideal para identificar músicas com o mesmo padrão sonoro, mesmo que com intensidades diferentes. Já a distância euclidiana penaliza essas diferenças de magnitude, o que pode ser enganoso para perfis musicais

Referências e Links

https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data
https://medium.com/pizzadedados/kmeans-e-metodo-do-cotovelo-94ded9fdf3a9
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.maxabs_scale.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html
https://scikit-learn.org/stable/index.html
https://htmlpreview.github.io/?https://github.com/DaviMattos14/obsidian_notebook/blob/main/Computa%C3%A7%C3%A3o%20Cient%C3%ADfica%20e%20An%C3%A1lise%20de%20Dados/Trabalho%20Final/Trabalho_Final.html