**Análise de Dados de Streaming do Spotify**

**Tomás Torre**

**09/2025**

1. CONTEXTO

O projeto analisa dados de streaming do Spotify, incluindo agregações, métricas de consumo, padrões temporais, engajamento e retenção, inicialmente utilizando Python (para fins de treinamento), quanto depois foi traduzido para PostgreSQL, com visualização em Power BI, resultando nos mesmos dados em ambas as análises, mas com metodologias diferentes.

Este projeto tem como objetivo explorar e analisar o histórico de streaming do Spotify, utilizando técnicas de engenharia de dados, análise exploratória, modelagem preditiva e análise de sentimentos.

A análise busca responder perguntas como:

* Quais padrões de escuta podem ser identificados ao longo do tempo?
* Quais dispositivos e plataformas são mais utilizados?
* Qual a previsão de consumo dos usuários baseada no histórico de dados?
* Quais músicas têm maior probabilidade de serem ouvidas novamente?
* Quais comportamentos podem indicar churn?
* Quais são as preferências e rejeições dos usuários?
* Podcasts e audiobooks são produtos viáveis e consumidos pelos usuários?

2. INTEGRAÇÃO DE PYTHON, SQL E POWER BI

A análise dos dados de streaming foi inicialmente conduzida em **Python**, o que permitiu realizar cálculos agregados, construir modelos preditivos e probabilísticos, e aplicar técnicas de clusterização. Modelos de **previsão com XGBoost** foram utilizados para estimar padrões futuros de consumo musical, enquanto **Random Forest** foi empregado para calcular probabilidades de eventos específicos com base em múltiplas árvores de decisão. Além disso, algoritmos de **clusterização do Scikit-Learn** permitiram agrupar usuários ou músicas com comportamentos semelhantes, possibilitando segmentações mais detalhadas.

Posteriormente, os mesmos cálculos e agregações foram traduzidos para **SQL** sobre a base armazenada no **PostgreSQL**, com ingestão dos dados realizada via Python, utilizando bibliotecas como ***sqlalchemy*, *os*** e ***datetime***, para empilhar e consolidar arquivos JSON do Spotify em uma tabela central, e, posteriormente, usando **PowerBI** como ferramenta de DataViz.

3. API SPOTIFY - Obtenção de Gêneros e Popularidade dos Artistas

3.1 OBJETIVO

Este script tem como finalidade extrair informações de gêneros musicais e popularidade de artistas do Spotify utilizando a API oficial (**spotipy**). Em vez de executar a API todas as vezes, os dados já processados podem ser carregados de um arquivo Excel para economizar tempo, visto que a execução completa demorou aproximadamente 66 minutos para 11.103 artistas.

* 1. FUNCIONAMENTO

As bibliotecas utilizadas neste script foram as seguintes:

* **Spotipy**: Biblioteca Python para interação com a API do Spotify, permitindo consultar informações sobre artistas, álbuns, faixas e playlists.
* **pandas**: Biblioteca para manipulação e análise de dados em estruturas como DataFrames, facilitando filtragem, agregação e exportação de dados.
* **time**: Módulo da biblioteca padrão do Python usado para controlar pausas ou atrasos na execução do código, útil para respeitar limites de requisições.
* **tqdm**: Biblioteca que fornece barras de progresso visuais para loops, permitindo acompanhar o avanço de tarefas demoradas.

A autenticação é realizada via **SpotifyClientCredentials**, criando o objeto sp para interação com a API do Spotify.

A API limita a quantidade de artistas que podem ser consultados por requisição. Por isso, o script processa os dados em **lotes de 50 artistas**, incluindo tratamento de **rate limit (HTTP 429)**. Quando esse limite é atingido, o script aguarda o tempo recomendado no cabeçalho Retry-After antes de tentar novamente. Para evitar sobrecarga, também é adicionada uma pausa de **0,2 segundos entre lotes normais**.

Após a execução, os resultados foram salvos em **arquivo Excel**, evitando requisições excessivas e o tempo elevado de processamento dos requests. Posteriormente, este arquivo Excel é recarregado no código para utilização posterior.

**Observações importantes:**

1. É necessário ter as credenciais corretas do Spotify e a biblioteca spotipy instalada.
2. O valor de **popularidade (popularity)** fornecido pelo Spotify varia de 0 a 100, refletindo o interesse atual do público pelo artista.

4. BASES DE DADOS

4.1 ORIGEM

Os dados utilizados neste projeto são **históricos reais do Spotify**, exportados diretamente das configurações da plataforma. Isso garante que as análises reflitam o comportamento verdadeiro de cada usuário e permite que qualquer pessoa replique o estudo com seu próprio perfil.

Embora seja possível usar a **API do Spotify**, optou-se pelos arquivos JSON para **praticar o processamento de dados JSON** e **ampliar o dataset com amigos**, sem necessidade de acessar suas contas. Essa abordagem também demonstra como o projeto poderia ser escalável para análises em larga escala, inclusive em contextos profissionais, mantendo a privacidade dos usuários.

4.2 TRATAMENTO DOS DADOS E BASE FINAL

A seguir, são apresentados os principais passos realizados no tratamento da base de dados, organizados em tópicos com explicações detalhadas:

* **Importação e combinação de arquivos JSON**
  + Todos os arquivos exportados do Spotify foram carregados por usuário e combinados em um único DataFrame.
  + Cada registro recebeu colunas adicionais: source\_file (nome do arquivo de origem) e pessoa (identificador do usuário).
  + Isso permite rastrear a origem dos dados e identificar facilmente o usuário correspondente.
* **Conversão e extração de informações temporais**
  + O campo de timestamp (ts) foi convertido para datetime para facilitar análises temporais.
  + Foram extraídas informações como ano, mês, dia, hora, dia da semana e um campo year\_month para agregações mensais.
  + Essa estrutura facilita a análise de padrões de consumo ao longo do tempo, tanto individual quanto coletivamente.
* **Categorização e transformação de colunas**
  + A plataforma de reprodução foi categorizada em Phone, Computer - online, Computer - desktop, TV e Others.
  + A duração das reproduções foi convertida de milissegundos para segundos e minutos.
  + O tipo de conteúdo foi classificado em Música, Podcast e Audiobook.
* **Remoção de colunas desnecessárias**
  + Colunas que não contribuem para análise, como informações de IP, país, modos offline e de incógnito, e detalhes específicos de episódios ou audiobooks, foram eliminadas.
  + Isso ajuda a reduzir ruído e facilita a manipulação do DataFrame.
* **Definição do intervalo temporal comum**
  + Para comparações consistentes entre usuários, foi definido um período comum:
    - **Data inicial:** maior valor mínimo de timestamp entre os usuários, ajustado para o início do mês seguinte.
    - **Data final:** menor valor máximo de timestamp entre os usuários.
  + O DataFrame final contém apenas registros dentro desse intervalo, garantindo consistência nas análises comparativas.
* **Disponibilidade da base final**
  + O resultado é uma base limpa, estruturada e escalável, pronta para análises exploratórias, visualizações e modelagem preditiva.
  + Permite análises individuais de um usuário ou agrupadas de múltiplos usuários, mantendo integridade e consistência dos dados.
  + Qualquer pessoa pode aplicar essa metodologia aos seus próprios dados exportados do Spotify, tornando o projeto replicável e escalável.

4.3 METADADOS

* **Origem:** JSON files (per user) > Python (pandas df) > Banco PostgreSQL > schema spotify.
* **Tabelas utilizadas:** *hist\_streaming*: histórico detalhado de reproduções do usuário, incluindo informações sobre músicas, álbuns, artistas, duração de reprodução, skips, shuffle, offline/online e plataforma.
* **Colunas principais:**
  + pessoa: identificador do usuário.
  + ts: timestamp de início da reprodução.
  + minutes\_played: duração em minutos da reprodução.
  + master\_metadata\_track\_name: nome da música.
  + master\_metadata\_album\_album\_name: nome do álbum.
  + master\_metadata\_album\_artist\_name: nome do artista.
  + tipo\_conteudo: tipo de conteúdo (música ou podcast).
  + shuffle: flag indicando se o shuffle estava ativo.
  + offline: flag indicando se a reprodução foi offline.
  + platform\_category: plataforma utilizada (ex.: desktop, mobile, web).
  + skipped: flag indicando se a faixa foi pulada.
* **Período de análise:** Varia por consulta, podendo cobrir desde o início do histórico do usuário até o presente, com filtros específicos por ano ou mês, porém, o intervalo utilizado (comum entre todos os usuários) é **entre 2020-10-01 até 2025-08-23**.

Uma imagem contendo Gráfico

O conteúdo gerado por IA pode estar incorreto.

5. INGESTÃO DE DADOS NO POSTGRESQL

O objetivo desta etapa foi ingerir os dados coletados e tratados via Python em um banco de dados PostgreSQL, garantindo que os dados pudessem ser consultados e analisados posteriormente de forma estruturada. Para isso, foi utilizado o *schema* chamado **spotify**, que precisa existir previamente no banco, pois o processo de ingestão cria apenas as tabelas, não o schema.

Para salvar os dados, foi criada uma função que recebe o DataFrame, o nome da tabela e algumas opções, como substituir a tabela existente ou adicionar dados a ela. Além disso, foi adicionada uma coluna com timestamp da carga, permitindo identificar quando cada registro foi inserido.

O processo foi utilizado para dois conjuntos de dados principais:

O **histórico de streaming já tratado (df\_combined),** contendo informações detalhadas sobre o consumo musical, e os **dados de artistas (df\_artistas),** contendo informações como gêneros e popularidade no Spotify.

Após a execução, os dados ficaram armazenados nas tabelas correspondentes dentro do schema spotify, permitindo consultas futuras sem necessidade de reprocessar os arquivos originais.

6. BIBLIOTECAS UTILIZADAS - PYTHON

Na análise realizada, diversas bibliotecas Python foram empregadas, cada uma voltada a funções específicas dentro do pipeline de dados, modelagem e visualização:



7. FUNÇÕES UTILIZADAS - SQL

A análise dos padrões de consumo musical utilizou **funções de agregação** para calcular totais, médias e extremos, além de **funções de tratamento de strings e datas** para padronização e agrupamento temporal. **Window functions** permitiram calcular médias móveis, streaks e rankings, enquanto **CTEs** organizaram consultas complexas, como filtragem de dados, métricas de retenção, identificação de novos artistas e cálculo de índices ponderados. Essa abordagem metodológica possibilitou capturar de forma robusta os padrões de escuta e engajamento dos usuários.



8. ANÁLISES

8.1 DESCRITIVA

**0) Testes iniciais**

* Verifica se existem dados para o usuário específico.
* Serve como teste de funcionamento do banco.

**1) Metadados da Tabela**

* Lista as colunas da tabela spotify.hist\_streaming e seus tipos de dados.
* Permite inspeção inicial da estrutura dos dados.

**2) Mínimo e máximo de ts por pessoa**

* Retorna a primeira e a última data de streaming registrada por usuário.
* Auxilia na análise do período de dados disponíveis.

**3) Intervalo mínimo usado na análise**

* Define o intervalo comum entre todos os usuários, garantindo comparabilidade.
* Calcula o maior mínimo e o menor máximo de datas.

**4) Filtragem para intervalo comum**

* Ajusta a base para conter apenas dados dentro do intervalo definido.
* Filtra para o usuário tomas.

**5) Média diária de minutos escutados**

* Calcula total diário de minutos escutados.
* Calcula média por dia do mês e por ano.
* Útil para identificar padrões de consumo diários.

**6) Padrões de escuta em 2025**

* Análise por **dia do mês**, **dia da semana**, e **hora do dia**.
* Divide horas entre dias de semana e finais de semana.
* Permite entender hábitos de escuta detalhados.

**7) Tendência de escuta ao longo do tempo**

* Agrega minutos por dia e converte para horas e dias.
* Calcula médias móveis de 3 meses para suavizar tendências.
* Serve para análise de séries temporais.

**8) Comparação 2025 x 2024 por mês**

* Soma minutos por mês para 2024 e 2025.
* Calcula variação percentual e médias mensais.
* Permite observar crescimento ou queda no consumo.

**9) Top 10 Artistas, Álbuns e Músicas**

* Agrega minutos tocados por artista, álbum e música.
* Ordena por total de minutos e limita ao top 10.
* Pode ser filtrado por ano.

**10) Distribuição das músicas mais escutadas**

* Cria conjuntos de top 20 músicas: **geral**, **2025**, e **último mês fechado**.
* Útil para gráficos de dispersão e análise de popularidade.

**11) Metodologia Top 100 – Índice ponderado**

* Normaliza minutos e quantidade de plays.
* Calcula índice final combinando tempo e quantidade.
* Ordena pelo índice e retorna top 100 músicas.

**12) Sessões de escuta**

* Define sessão como sequência de músicas com intervalo ≤ 30 minutos.
* Calcula duração, número de faixas e hora de início.
* Analisa sessões por hora do dia, dia da semana e número médio de faixas.

**13) Taxa de skip**

* Calcula skip rate por música e por artista.
* Limita análises para músicas com reproduções mínimas (≥5) e artistas com ≥10 plays.
* Retorna top 20 com maior taxa de skip.

**14) Uso de Shuffle e Offline**

* Analisa proporção de faixas tocadas com shuffle ativado/desativado.
* Analisa proporção de faixas tocadas offline.
* Dados retornam percentuais para gráficos de pizza.

**15) Análise por plataforma**

* Mostra proporção de faixas tocadas por plataforma.
* Analisa participação percentual das plataformas ao longo do tempo.
* Permite identificar preferência por apps ou dispositivos.

**16) Faixas únicas vs repetidas**

* Calcula proporção de músicas escutadas uma única vez vs repetidas.
* Também gera análise mensal para acompanhar evolução ao longo do tempo.

**17) Identificação de músicas e artistas novos**

* Determina primeiro mês de aparição de cada música e artista.
* Conta músicas novas por mês e calcula proporção de músicas de artistas novos.
* Permite identificar tendências de novidade no consumo.

**18) Diversidade musical**

* Calcula índice de **Shannon** mensal para medir diversidade de artistas ou faixas.
* Mostra número de artistas únicos e concentração de escuta.

**19) Engajamento com podcasts**

* Soma minutos escutados de podcasts por mês.
* Calcula proporção do tempo de podcast em relação ao consumo total.

**20) Mudança de preferências e concentração de favoritos**

* Identifica top 20 artistas mais escutados em um período.
* Calcula proporção de minutos desses favoritos em relação ao total mensal.
* Permite ver se o usuário se concentra em poucos artistas.

**21) Retenção e frequência de uso**

* Conta dias únicos de escuta.
* Calcula número de gaps (dias sem escuta) e média de dias por gap.
* Identifica maior streak de dias consecutivos com reprodução.
* Fornece métricas de engajamento e hábitos de uso ao longo do tempo.

**22) Análise de preenchimento de estilos e gêneros (resumido)**

* Classifica músicas em preenchidas ou nulas/em branco para estilo e gênero.
* Soma os minutos tocados por categoria.
* Visualiza a distribuição com gráficos de pizza, mostrando a proporção de tempo de escuta entre músicas com dados completos e incompletos.
* Objetivo: entender quanto do tempo de escuta está associado a músicas com informações completas de estilo e gênero**.**

**23) Análise por gênero musical e evolução por estilo ao longo do tempo**

* **Top 10 por estilo e primary\_style:**
  + Agrupa os minutos de escuta por estilo e primary style.
  + Identifica os 10 estilos/gêneros mais ouvidos.
  + Visualiza com gráficos de barras horizontais, mostrando minutos totais ouvidos.
* **Evolução ao longo do tempo (Top 5 estilos):**
  + Agrupa minutos de escuta por mês e por estilo.
  + Seleciona os 5 estilos mais ouvidos no total.
  + Plota a evolução temporal dos top 5 estilos para:
    - Geral (todos os anos disponíveis).
    - Ano específico (2025), destacando tendências recentes.
* **Top 5 vs Resto vs Nulos:**
  + Categoriza os estilos em Top 5, Resto e Nulos/Em branco.
  + Soma minutos por mês e grupo, criando séries para cada categoria.
  + Visualiza a evolução com:
    - Linhas sobrepostas, destacando o volume de cada grupo ao longo do tempo.
    - Gráfico de barras 100% empilhadas, mostrando a proporção mensal de minutos tocados entre Top 5, Resto e Nulos.
* **Objetivos da análise:**
  + Identificar quais estilos e gêneros dominam a escuta total.
  + Acompanhar tendências ao longo do tempo e mudanças no comportamento de consumo.
  + Avaliar a importância dos top 5 estilos em relação ao resto e à presença de dados nulos.

8.2 PREDITIVA

8.2.1 **XGBoost** - Previsão de escuta futura

* **Preparação dos dados**:
  + Converte a coluna de timestamp para datetime.
  + Calcula minutos de reprodução a partir de milissegundos.
* **Agregação diária**:
  + Soma minutos de escuta por dia.
  + Remove o mês atual incompleto para evitar distorções na previsão.
* **Criação de variáveis explicativas (features)**:
  + day\_index: índice sequencial de cada dia.
  + day\_of\_week: dia da semana (0=segunda, 6=domingo).
  + month: mês do ano.
* **Separação de features e target**:
  + Features (X): day\_index, day\_of\_week, month.
  + Target (y): minutes\_played.
* **Escalonamento dos dados**:
  + Normaliza as features usando StandardScaler para melhorar o desempenho do modelo.
* **Treinamento do modelo**:
  + Modelo XGBRegressor treinado com os dados diários escalonados.
* **Previsão para o futuro (~1 ano)**:
  + Cria índices e datas para os próximos 365 dias.
  + Calcula dia da semana e mês para cada data futura.
  + Transforma essas features usando o mesmo scaler aplicado ao treino.
  + Gera previsões de minutos diários.
* **Agregação mensal**:
  + Soma os minutos previstos por mês, transformando os períodos em timestamps mensais.
* **Visualização**:
  + Plota a previsão mensal de minutos de escuta para os próximos 6 meses.
  + Configura título, labels, grid e legenda para facilitar interpretação.

8.2.3 **Random Forest** - Classificação de músicas por probabilidade de serem ouvidas novamente

* **Criação da variável alvo**:
  + listened\_again indica se a música foi ouvida mais de uma vez pelo usuário.
  + É gerada a partir de duplicações de faixas considerando nome da música e artista.
* **Seleção de features**:
  + Inclui características como seconds\_played, skipped, shuffle, time, platform\_category, reason\_start, reason\_end e tipo\_conteudo.
  + Essas features descrevem o comportamento de escuta e contexto da reprodução.
* **Codificação de variáveis categóricas**:
  + Variáveis do tipo objeto são transformadas em números usando LabelEncoder, permitindo que o modelo Random Forest trabalhe com elas.
* **Treinamento do modelo**:
  + Divisão em treino e teste (80% treino / 20% teste).
  + Modelo RandomForestClassifier com 100 árvores (n\_estimators=100).
* **Avaliação do modelo**:
  + **Precision**: proporção de acertos entre as previsões feitas para cada classe.
    - Classe 0 (não ouvida novamente): apenas 18% das previsões corretas.
    - Classe 1 (ouvida novamente): 93% das previsões corretas.
  + **Recall**: proporção de acertos entre todos os exemplos reais de cada classe.
    - Classe 0: modelo identificou corretamente 12% dos casos reais.
    - Classe 1: 96% dos casos reais identificados corretamente.
  + **F1-score**: média harmônica entre precision e recall, útil em situações de **desbalanceamento de classes**.
  + **Support**: número de exemplos reais de cada classe no conjunto de teste.
  + **Acurácia geral**: 90%, mas enganosa devido ao desbalanceamento de classes, pois o modelo acerta principalmente a classe majoritária (ouvida novamente) e ignora a minoritária (não ouvida novamente).
* **Problema identificado**:
  + Há um **desbalanceamento significativo entre classes**, com muito mais exemplos de músicas ouvidas novamente (classe 1).
  + O modelo tende a priorizar a classe majoritária, prejudicando a identificação da classe minoritária (não ouvida novamente).

9. CÓDIGO SQL E POWER BI

Após recriar todas as análises de forma descritiva no SQL (PostgreSQL), inseri as bases correspondentes no banco e apliquei cada código no Power BI. Esse processo permitiu treinar tanto o desenvolvimento de consultas SQL quanto a criação de visualizações de dados no Power BI, utilizando Native Query para integrar diretamente as análises do banco às visualizações.

CONCLUSÃO

No projeto, consolidei e tratei arquivos JSON em lote, criando uma base unificada e transformando colunas para análise. Integrei dados externos através da API oficial do Spotify, enriquecendo informações sobre artistas e gêneros. Realizei análises descritivas estáticas, dinâmicas e temporais, explorando padrões de consumo, preferências, tendências e comportamento musical do usuário.

Apesar de neste relatório estar mostrando apenas uma pessoa, o código foi desenvolvido para rodar para todos os usuários simultaneamente, caso necessário. Se estivesse trabalhando para uma empresa como o Spotify, seria possível obter informações de comportamento de grupos de usuários por tipo de gosto, região, padrões de escuta, probabilidades de churn, entre outras métricas, além de realizar análises de previsão. Todo esse trabalho pode ser feito de forma multiplataforma, utilizando Python, SQL e Power BI.