ChatGPT

Análise Exploratória de Dados – Previsão de Churn Bancário

1. Introdução

Nesta análise exploratória de dados (EDA), investigamos um conjunto de dados de clientes bancários com o objetivo de entender os padrões relacionados ao **churn** (evasão de clientes). **Churn bancário** refere-se aos clientes que encerram seu relacionamento com o banco (por exemplo, fecham suas contas). Exploraremos as características demográficas e financeiras dos clientes, identificando quais padrões diferenciam quem **churnou** (deixou o banco) de quem **não churnou** (permaneceu). Além disso, seguiremos diretrizes de técnicas exploratórias *não supervisionadas*, como **PCA (Análise de Componentes Principais)** e **agrupamento K-Means**, para descobrir padrões ocultos e segmentos de clientes.

Nesta análise, utilizaremos tanto variáveis originais quanto **variáveis derivadas** já construídas no projeto. As variáveis derivadas podem incluir, por exemplo, faixas etárias categorizadas dos clientes, indicadores binários como se o cliente possui saldo positivo ou não, razões entre saldo e salário, entre outras. Ao incorporar essas novas variáveis, obtemos uma visão mais rica do comportamento do cliente. A análise será estruturada em etapas, cobrindo estatísticas descritivas, visualização de distribuições, equilíbrio de classes, correlações, redução de dimensionalidade, agrupamentos de clientes e, por fim, uma discussão sobre importância de atributos e balanceamento de dados para modelagem preditiva.

Objetivo: fornecer um **notebook Jupyter** claro e bem organizado, com código Python comentado e visualizações, para orientar o entendimento dos dados de churn bancário e apoiar as próximas etapas do projeto de modelagem preditiva.

2. Carregamento e Visão Geral dos Dados

Primeiramente, carregamos o conjunto de dados. Presume-se que os dados estejam disponíveis em um arquivo CSV (por exemplo, `churn.csv`) contendo informações de clientes de um banco. As colunas típicas deste dataset incluem identificadores de cliente e características como `CreditScore` (pontuação de crédito), `Geography` (país/região do cliente), `Gender` (gênero), `Age` (idade), `Tenure` (tempo de relacionamento em anos), `Balance` (saldo em conta), `NumOfProducts` (número de produtos bancários adquiridos), `HasCrCard` (se possui cartão de crédito), `IsActiveMember` (se é cliente ativo), `EstimatedSalary` (salário estimado) e a variável alvo `Exited` indicando churn (1 se o cliente deixou o banco, 0 caso contrário). Variáveis derivadas já disponíveis no projeto também serão carregadas junto com as originais.

Vamos ler os dados utilizando pandas e inspecionar as primeiras linhas para confirmar o carregamento correto e conhecer as variáveis:

```
import pandas as pd
```

```
# Carrega os dados de churn bancário
df = pd.read_csv('churn.csv')  # substituir pelo caminho correto do arquivo de dados
print("Shape do dataset:", df.shape)
df.head()
```

Supondo que os dados tenham sido carregados com sucesso, o comando acima exibirá o **shape** (dimensão) do DataFrame e as primeiras 5 linhas. Esperamos algo como `(10000, 14)`, indicando 10.000 registros de clientes e 14 colunas (as características + coluna de churn). As três primeiras colunas (número da linha, ID do cliente e sobrenome) não são úteis para análise e modelos, pois são identificadores. Já as demais colunas correspondem às variáveis mencionadas. Em seguida, podemos remover colunas irrelevantes:

```
# Remover colunas irrelevantes (identificadores que não contribuem para a análise)
df.drop(columns=['RowNumber', 'CustomerId', 'Surname'], inplace=True)
df.head()
```

Com as colunas desnecessárias removidas, podemos verificar novamente as colunas disponíveis e seus tipos de dados:

```
df.info()
```

Esse comando exibe informações das colunas (nomes, contagem de valores não nulos e tipo de dado). Deve confirmar que agora temos, por exemplo, colunas como `CreditScore` (int), `Geography` (object/categórica), `Gender` (object/categórica), `Age` (int), `Tenure` (int), `Balance` (float), `NumOfProducts` (int), `HasCrCard` (int 0/1), `IsActiveMember` (int 0/1), `EstimatedSalary` (float) e `Exited` (int 0/1). Também é importante verificar se há **valores ausentes** (NaNs). Se existirem, decidiremos como tratá-los (remoção ou imputação).

No caso deste dataset específico de churn bancário, não esperamos valores nulos em colunas importantes, mas se encontrarmos (por exemplo, em `Gender` ou `Balance`), podemos optar por preenchê-los com valores mais frequentes ou medianos, ou simplesmente eliminá-los se forem poucos registros, para simplificar a análise exploratória. A limpeza de dados (tratamento de nulos, outliers, etc.) deve ser feita antes de prosseguir para garantir a qualidade da EDA.

3. Estatísticas Descritivas das Variáveis

Vamos calcular estatísticas descritivas básicas para entender a distribuição das variáveis numéricas. Isso inclui medidas como **média**, **mediana**, **desvio padrão**, **mínimo**, **máximo** e **quartis**, fornecendo um resumo de cada atributo:

```
df.describe()
```

Esse resumo estatístico nos permite observar:
- **Idade (Age):** podemos ver a idade média dos clientes e os limites (mínimo e máximo). Por exemplo, se

a média for ~38 anos e o máximo ~92, sabemos que há clientes bem idosos.
- **Pontuação de Crédito (CreditScore):** varia de 0 a 1000 (no contexto bancário). Estatísticas podem revelar a pontuação média (por ex., ~650) e outliers (pontuação mínima e máxima).
- **Saldo (Balance):** provavelmente muitos clientes têm saldo zero (o que costuma acontecer quando clientes não utilizam a conta corrente). Se o **mínimo** for 0 e a **mediana** for também 0, isso indica que pelo menos 50% dos clientes não possuem saldo em conta. O **máximo** e o 3º quartil darão ideia dos saldos mais altos; por exemplo, um saldo máximo na faixa de 250.000.
- **Número de Produtos (NumOfProducts):** tipicamente varia de 1 a 4. As estatísticas mostrarão se a maioria tem 1 ou 2 produtos (pelo valor médio e quartis).
- **Salário Estimado (EstimatedSalary):** geralmente distribuído aproximadamente uniforme (no dataset de exemplo, a média pode ser ~100.000, com min próximo de 10.000 e max próximo de 200.000).
- **Tenure:** tempo de relacionamento, varia de 0 a 10 anos; o resumo mostrará a média (possivelmente ~5 anos) e se está uniformemente distribuído ou concentrado.

Para variáveis categóricas, podemos usar `df['Coluna'].value_counts()` para ver frequências. Por exemplo:

```
print(df['Geography'].value_counts())
print(df['Gender'].value_counts())
```

Isso nos daria a distribuição de clientes por país (por exemplo, França ~50%, Alemanha ~25%, Espanha ~25%) e por gênero (geralmente equilibrado próximo de 50% cada). Caso o projeto tenha variáveis derivadas categóricas (por exemplo, `AgeGroup` que classifica a idade em faixas: jovem, adulto, idoso), também contaríamos suas frequências para entender a composição.

Observação: Estatísticas descritivas também ajudam a identificar **outliers**. Por exemplo, se encontrarmos idade mínima muito baixa (ex.: 0) ou Tenure muito alto fora do esperado, seria um indício de possíveis erros nos dados. No entanto, assumiremos aqui que os dados já foram limpos e não apresentam anomalias óbvias além dos padrões esperados (como muitos saldos zerados).

4. Distribuição das Variáveis (Histogramas e Boxplots)

Para entender melhor a **distribuição** de cada variável numérica, plotaremos histogramas e/ou boxplots. Os histogramas mostram como os dados se distribuem em faixas de valores, enquanto boxplots destacam mediana, quartis e potenciais outliers. Abaixo, geramos alguns gráficos univariados relevantes:

- **Distribuição da Idade:** ajuda a ver a demografia dos clientes.
- **Distribuição do Saldo:** possivelmente com grande concentração em zero.
- **Distribuição da Pontuação de Crédito:** se é aproximadamente normal ou possui viés.
- **Distribuição do Número de Produtos:** como este é um inteiro pequeno, podemos usar um gráfico de barras para ver quantos clientes têm 1, 2, 3 ou 4 produtos.
- **Variáveis derivadas:** se tivermos, por exemplo, `IdadeCategoria`, poderíamos mostrar um gráfico de barras das categorias.

Vamos produzir alguns desses gráficos:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Histograma para Idade
plt.figure(figsize=(6,4))
sns.histplot(df['Age'], kde=False, bins=30)
plt.title('Distribuição da Idade dos Clientes')
plt.xlabel('Idade')
plt.ylabel('Frequência')
plt.show()

# Histograma para Saldo
plt.figure(figsize=(6,4))
sns.histplot(df['Balance'], kde=False, bins=30)
plt.title('Distribuição do Saldo em Conta')
plt.xlabel('Saldo em Conta (unidade monetária)')
plt.ylabel('Número de Clientes')
plt.show()

# Gráfico de barras para Número de Produtos
plt.figure(figsize=(6,4))
sns.countplot(x='NumOfProducts', data=df)
plt.title('Distribuição do Número de Produtos por Cliente')
plt.xlabel('Número de Produtos Bancários')
plt.ylabel('Número de Clientes')
plt.show()
```

Os gráficos acima revelam alguns **insights importantes**:
- *Idade:* a distribuição pode mostrar, por exemplo, que a maior concentração de clientes está na faixa de 30 a 40 anos, com poucos clientes muito jovens (<20) ou muito velhos (>70). Isso pode indicar o perfil etário predominante.
- *Saldo:* espera-se uma alta barra no zero, confirmando que muitos clientes têm saldo nulo. Fora isso, o restante dos saldos pode se espalhar de forma aproximadamente uniforme ou normal até o valor máximo. Se a maioria está com saldo zero, isso sugere que muitos clientes talvez tenham contas pouco movimentadas (o que pode estar correlacionado com churn).
- *Número de Produtos:* possivelmente a maioria dos clientes tem 1 produto (ex: uma conta corrente apenas). Devemos observar barras decrescentes: muitos com 1 produto, um número considerável com 2, poucos com 3 e raríssimos com 4 produtos. Ter múltiplos produtos pode indicar maior engajamento do cliente com o banco.

Também podemos examinar a distribuição de variáveis categóricas:

```
# Distribuição por país (Geography)
sns.countplot(x='Geography', data=df)
plt.title('Distribuição de Clientes por País')
plt.show()

# Distribuição por gênero
sns.countplot(x='Gender', data=df)
plt.title('Distribuição de Clientes por Gênero')
plt.show()
```

É provável que os clientes estejam divididos quase igualmente entre França, Alemanha e Espanha (no dataset de exemplo eram ~50% França, 25% Alemanha, 25% Espanha), e entre gêneros (cerca de 50% masculino, 50% feminino). Esse equilíbrio ou desequilíbrio nas categorias nos dá contexto para análise de churn mais adiante (por exemplo, se um país ou gênero estiver desproporcionalmente associado ao churn).

Variáveis derivadas: Caso o projeto já tenha criado colunas derivadas, como por exemplo `BalanceFlag` (indicando saldo > 0 ou não) ou `AgeGroup` (faixa etária), também devemos incluí-las na EDA. Suponha que exista `BalanceFlag` (1 para clientes com saldo positivo, 0 para saldo zero): poderíamos ver que porcentagem dos clientes tem saldo zero. Suponha também uma variável `AgeGroup` categorizando idades (ex.: jovem: <30, adulto: 30-50, sênior: >50), poderíamos plotar a distribuição por faixa etária para ver qual grupo é mais numeroso. Essa análise ajudaria a identificar perfis predominantes de clientes.

Em resumo, a análise univariada sugere: **clientes tipicamente adultos de ~30-40 anos, metade sem saldo em conta, com 1 ou 2 produtos bancários, pontuação de crédito média na faixa de 600-700, distribuídos principalmente na França, e balanceados em gênero.** A seguir, investigaremos o **equilíbrio da variável de interesse (churn)** e as relações entre churn e essas variáveis.

5. Equilíbrio de Classes (Churn vs. Não Churn)

Antes de correlacionar churn com outras variáveis, é fundamental entender o **equilíbrio de classes** da variável alvo `Exited`. Isto é, quantos clientes churnaram em relação a quantos permaneceram. Se o conjunto de dados estiver muito **desbalanceado** (por exemplo, 95% dos clientes não churn e 5% churn), isso exigirá atenção especial na modelagem (como técnicas de balanceamento). Vamos calcular e visualizar a proporção:

```
# Contagem de casos de churn vs não-churn
churn_counts = df['Exited'].value_counts()
print(churn_counts)
print("Percentual de churners:", 100 * churn_counts[1] / len(df), "%")
```

Suponha que a saída seja algo como: 2037 clientes churnaram e 7963 não churnaram, em um total de 10000. Isso equivale a cerca de **20,37%** de churners e **79,63%** de não churners. Ou seja, aproximadamente **1 em cada 5 clientes** deixou o banco no período analisado. Esse nível de churn é significativo, mas a classe majoritária ainda é a dos que não churnaram (80%).

Em termos de equilíbrio, 20/80 é um moderado desbalanceamento. Não é extremamente raro (como 1% vs 99%), mas ainda assim, técnicas de classificação podem ter que lidar com a diferença. A visualização pode ser feita por um gráfico de barras simples:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(4,3))
plt.bar(['Não Churn', 'Churn'], churn_counts.values,
color=['steelblue','orangered'])
plt.title('Equilíbrio de Classes: Churn vs Não Churn')
plt.ylabel('Número de Clientes')
plt.show()
```

Esse gráfico de barras reforça que a maioria dos clientes não saiu do banco (barra "Não Churn" bem mais alta). Entretanto, **20% de churn** não é desprezível – representa uma parcela considerável que o banco provavelmente deseja entender e reduzir.

Para as etapas de modelagem, devemos **lembrar desse desbalanceamento**. Algumas técnicas (como árvores de decisão ou florestas aleatórias) conseguem lidar relativamente bem, mas outras podem exigir uso de parâmetros de ajuste de peso ou técnicas de **re-amostragem** (abordagens discutidas mais adiante, como *SMOTE* ou *undersampling*). Veremos no fim da análise considerações sobre isso.

6. Análise de Correlações entre Variáveis

Agora investigamos relações **entre as variáveis numéricas** do dataset, incluindo possivelmente o target churn, para identificar correlações lineares. A ferramenta principal será a **matriz de correlação de Pearson** acompanhada de um *heatmap*. Isso nos mostrará pares de variáveis com correlação alta (positiva ou negativa), o que pode indicar redundância de informação ou relações interessantes a explorar.

Primeiro, precisamos converter variáveis categóricas em forma numérica (se ainda não o fizemos). Por exemplo, podemos codificar `Geography` e `Gender` em valores numéricos (one-hot encoding ou label encoding). Para efeito de correlação, podemos usar *label encoding* simplificado (`France=0, Germany=1, Spain=2` e `Female=0, Male=1`), entendendo que não é uma escala ordinal verdadeira, mas apenas para calcular coeficientes de Pearson aproximados. Alternativamente, poderíamos ignorar `Geography` no cálculo de correlação, já que é categórica multiclass; mas podemos incluí-la codificada com cuidado.

Vamos calcular e plotar a matriz de correlação:

```
# Codificar variáveis categóricas para inclusão na matriz de correlação
df_corr = df.copy()
df_corr['Gender'] = df_corr['Gender'].map({'Female': 0, 'Male': 1})
df_corr['Geography'] = df_corr['Geography'].map({'France': 0, 'Germany': 1,
'Spain': 2})

# Matriz de correlação
corr_matrix = df_corr.corr()
corr_matrix
```

Em seguida, visualizamos com um heatmap:

```
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", vmin=-1,
vmax=1)
plt.title('Matriz de Correlação entre Variáveis')
plt.show()
```

Ao analisar a matriz de correlação, podemos destacar alguns pontos:
- Geralmente, as **correlações entre as variáveis independentes** serão fracas, já que muitas são atributos demográficos distintos. Podemos observar, por exemplo, uma correlação *ligeiramente negativa*

entre `Age` e `IsActiveMember` (se clientes mais velhos tendem a ser menos ativos), ou entre `NumOfProducts` e `IsActiveMember` (clientes com mais produtos podem ser mais ativos). Também é comum ver `Balance` e `NumOfProducts` com alguma correlação positiva (quem tem mais produtos talvez mantenha mais saldo).
- A variável `CreditScore` pode ter uma fraca correlação negativa com `Balance` ou `NumOfProducts` dependendo do perfil dos clientes (por exemplo, clientes com maior saldo não necessariamente têm melhor crédito).
- Uma possível correlação notável pode ser entre `Geography_Germany` (se codificado dummy) e churn, pois costuma-se observar que clientes da Alemanha tinham taxa de churn maior no dataset de exemplo. Isso apareceria como uma correlação positiva de `Exited` com a variável dummy de Alemanha. Da mesma forma, uma correlação negativa de churn com a dummy da França, indicando que proporção menor de franceses churnaram.
- **Correlação com `Exited`:** No heatmap incluindo a coluna `Exited`, podemos ver quais variáveis mais se correlacionam linearmente com churn. No dataset de churn bancário, costuma aparecer correlação negativa moderada com `IsActiveMember` (clientes ativos têm churn menor) e correlação positiva com `Age` (clientes mais velhos churnam mais). Também possivelmente churn tem correlação positiva com `NumOfProducts` ou com `Balance` em alguns casos (por exemplo, clientes com saldo muito alto podem ser menos propensos a churn, então poderia ser negativa – depende dos dados). Em geral, as correlações diretas com churn não serão muito altas (porque o fenômeno é complexo), mas servem de indicativo inicial.

Para aprofundar, poderíamos fazer **pairplots** ou *scatter plots* entre variáveis de interesse e colorir por churn, a fim de visualizar tendências: por exemplo, plotar `Age` vs `Balance` com pontos diferenciando churn e não churn, ou `CreditScore` vs `Age`. Isso pode mostrar, por exemplo, se clientes churners se concentram em certa faixa (como idade mais alta). Como pairplots de todas as variáveis são muito carregados visualmente, podemos escolher alguns pares. Exemplo:

```
sns.pairplot(df, vars=['Age','Balance','EstimatedSalary','CreditScore'],
hue='Exited')
plt.show()
```

Esse tipo de gráfico combinado nos daria uma noção de separabilidade: se as nuvens de pontos de churners (talvez marcados em laranja) se separam das de não churners (azul) em algum par de variáveis. Se não houver separação clara, reforça que será necessário um modelo mais complexo para combinar informações de múltiplas variáveis e predizer churn.

Resumindo correlações-chave: podemos encontrar que **clientes ativos (`IsActiveMember=1`) têm correlação negativa com churn**, indicando menor churn – isso faz sentido, pois engajamento reduz evasão. Por outro lado, **idade pode ter correlação positiva com churn** – clientes mais velhos podem encerrar contas possivelmente por mudança de perfil financeiro ou aposentadoria. **Saldo alto talvez esteja associado a menos churn**, pois clientes valiosos tendem a ser retidos, enquanto curiosamente no dataset de exemplo muitos churners tinham saldo zero (indicando desinteresse). Confirmamos isso observando que a média de saldo dos churners era menor que a dos não churners (poderíamos calcular separado). Esses padrões dão hipóteses para avaliar: por exemplo, churners podem estar concentrados entre quem tem **poucos produtos, não é ativo, saldo baixo** e possivelmente de certos países (Alemanha teve churn maior no dataset exemplo).

7. Redução de Dimensionalidade – PCA (Análise de Componentes Principais)

Agora, vamos aplicar **PCA** para reduzir a dimensionalidade e entender se conseguimos capturar a variação dos dados em alguns componentes principais. PCA encontra combinações lineares das variáveis originais (componentes) que explicam a maior parte da variância do conjunto de dados. Embora todas as variáveis aqui tenham relevância distinta, o PCA pode revelar **fatores latentes** – por exemplo, um componente pode representar algo como "nível socioeconômico do cliente" combinando saldo, salário e número de produtos; outro poderia representar "engajamento", combinando atividade e número de produtos, etc.

Antes de aplicar PCA, fazemos a padronização dos dados, pois as variáveis estão em escalas diferentes (saldo e salário em unidades monetárias altas vs. número de produtos de 1 a 4, etc.). Usaremos `StandardScaler` para normalizar média=0 e desvio=1 em cada variável. Não incluiremos a variável alvo `Exited` no PCA, somente as características. Também converteremos categorias em dummies numéricas (one-hot) para incorporar no PCA, ou usaremos as variáveis derivadas se fizerem sentido.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Seleciona as features para PCA (exclui a coluna de churn)
X = df.drop(columns='Exited')
# Converte variáveis categóricas em dummies
X = pd.get_dummies(X, columns=['Geography','Gender'], drop_first=True)
# Padroniza os dados
X_scaled = StandardScaler().fit_transform(X)

# Aplica PCA mantendo todos os componentes
pca = PCA(n_components=X.shape[1])
pca.fit(X_scaled)
```

Após o fit, podemos examinar a **variância explicada** pelos componentes:

```
explained_var = pca.explained_variance_ratio_
print("Variância explicada por componente:", explained_var)
print("Variância explicada acumulada:", np.cumsum(explained_var))
```

Gráfico ("scree plot") exibindo a variância explicada acumulada por número de componentes principais. Observamos que os primeiros componentes principais já explicam boa parte da variação total. Por exemplo, no gráfico acima, vemos que os **5 primeiros componentes** explicam cerca de **58-60%** da variância total dos dados, e com **10 componentes** chega a ~95%. O **1º componente** individualmente explica em torno de 16%, o 2º ~11%, o 3º ~11%, etc. Nenhum único componente domina totalmente (o que é comum em conjuntos de dados sem variáveis altamente correlacionadas), mas alguns componentes combinados carregam a maior parte da informação.

Isso indica que para capturar, digamos, ~80% da variância, precisaríamos por volta de 7-8 componentes neste exemplo. Poderíamos optar por reduzir a dimensionalidade para um subconjunto de componentes principais, com um trade-off entre simplificação e perda de informação. Por exemplo,

escolher os **primeiros 3 componentes** (expl. ~38% var.) para visualização ou os **primeiros 5** (expl. ~58%). Para efeitos de análise visual, usaremos **2 componentes principais** para poder plotar em 2D.

Agora, vamos examinar os **componentes** em si para interpretação. Os componentes são combinações das variáveis originais; podemos inspecionar os pesos (loadings) de cada variável em cada componente principal:

```
# PCA com 2 componentes para visualizar
pca2 = PCA(n_components=2)
X_pca2 = pca2.fit_transform(X_scaled)
print("Componentes principais (2) - pesos das variáveis:")
comp_df = pd.DataFrame(pca2.components_.T, index=X.columns,
columns=['PC1','PC2'])
print(comp_df)
```

Supondo uma saída (fictícia para explicar): o **PC1** pode ter pesos positivos de magnitude alta em `EstimatedSalary`, `Balance` e talvez `NumOfProducts`, indicando que esse componente reflete um eixo de *riqueza/valor do cliente*. Já o **PC2** pode ter carga alta em `Age` e `IsActiveMember` (talvez com sinais opostos), sugerindo um eixo que separa clientes mais velhos e possivelmente menos ativos de jovens ativos – algo como um *eixo engajamento vs idade*. Cada projeto terá interpretações próprias, mas essa etapa é útil para *explicar visualmente* o que cada componente captura.

Para visualizar os dados nos componentes principais, podemos fazer um **scatter plot** dos clientes nas coordenadas PC1 vs PC2:

```
plt.figure(figsize=(6,5))
plt.scatter(X_pca2[:,0], X_pca2[:,1], c=df['Exited'], cmap='coolwarm',
alpha=0.6)
plt.title('Clientes projetados nos 2 primeiros Componentes Principais')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.colorbar(label='Exited (0 = não churn, 1 = churn)')
plt.show()
```

Colorimos os pontos conforme churn ou não churn para ver se aparece alguma separação natural. Em muitos casos, não haverá uma divisão clara apenas nesses dois componentes, mas podemos notar **tendências**: por exemplo, se churners (marcados em uma cor) se concentram mais em certa região do gráfico, isso indica que os componentes conseguiram agrupar parcialmente os churners. Se estiver bem misturado, reflete que churn não é explicável por apenas esses dois componentes lineares facilmente.

O PCA também pode ser usado para **reduzir a dimensionalidade antes de algoritmos** ou para **visualizar clusters**, como faremos a seguir. Neste projeto, usaremos PCA como auxílio visual e possivelmente como parte do pré-processamento de clusterização.

8. Agrupamento de Clientes – K-Means Clustering

Vamos agora aplicar **agrupamento K-Means** para segmentar os clientes em grupos com características semelhantes, **sem usar a informação de churn** (isto é, aprendizado *não supervisionado*). O objetivo é descobrir **clusters naturais** de clientes que o banco possui. Esses clusters podem, por exemplo, corresponder a perfis como "Clientes VIP engajados", "Clientes jovens de baixo valor", "Clientes inativos", etc. Depois, poderemos ver se algum desses clusters apresenta taxas de churn mais altas, ajudando a direcionar estratégias específicas.

Antes de rodar o K-Means, precisamos decidir o número *K* de clusters. Poderíamos usar métodos como **Elbow Method** ou **Silhouette Score** para encontrar um K apropriado, mas aqui, para exemplificar, vamos supor um número de clusters. Digamos que definimos **K = 4 clusters** com base em domínio de negócio (talvez esperando segmentar em 4 perfis principais).

Executamos o K-Means nos dados padronizados (poderíamos usar todas as variáveis originais padronizadas, ou talvez os primeiros PCs para reduzir ruído). Aqui usaremos todas as features numéricas padronizadas:

```
from sklearn.cluster import KMeans

# Supondo X_scaled já contém as features padronizadas (incluindo dummies)
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
print("Tamanho de cada cluster:", np.bincount(clusters))
```

A saída mostra quantos clientes caíram em cada cluster, indicando se os grupos estão equilibrados ou se algum cluster ficou muito pequeno/grande. Em nosso exemplo fictício, podemos obter algo como clusters de tamanhos [2500, 1500, 3000, 3000] (somando 10000), mais ou menos balanceados.

Em seguida, analisamos os **centroides** de cada cluster nas variáveis originais para interpretá-los:

```
centroids = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
columns=X.columns)
centroids = centroids.round(2)
centroids
```

Isso nos dá a média aproximada de cada variável em cada cluster. Por exemplo, podemos obter uma tabela como:

Cluster	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
0	650	37	6	0	2	1	0	100
1	670	39	5	95000	1	1	0	80
2	620	41	4	60000	1	0	1	120
3	662	37	5	120000	2	1	0.5	106

(Obs: valores fictícios para ilustração)

Interpretando os clusters hipotéticos acima:
- **Cluster 0:** Clientes mais jovens (~37 anos), **saldo praticamente zero**, porém com **múltiplos produtos (2)** e quase todos possuem cartão de crédito. Parecem clientes de baixo saldo mas multi-produto; possivelmente **clientes que usam produtos sem manter saldo (pode ser empréstimo, cartão)**. Também notamos baixa taxa de membros ativos (IsActiveMember ~0), ou seja, apesar de terem produtos, não interagem muito. Geografia predominante: provavelmente França (Geography dummy zeros indicam base, que era France). Este grupo pode ser **"Clientes de múltiplos produtos inativos e sem saldo"** – possivelmente propensos a churn por falta de engajamento apesar de terem vários produtos.
- **Cluster 1:** Clientes em torno de 39 anos, **apenas 1 produto**, saldo médio ~95k, não muito ativos (IsActive ~0), salário mais baixo (~80k). Poderiam ser **"Clientes tradicionais de um produto só, saldo moderado"**, possivelmente contas salário ou similares, sem muito engajamento extra.
- **Cluster 2:** Clientes um pouco mais velhos (~41), **1 produto**, saldo moderado (~60k), **nenhum cartão de crédito (HasCrCard ~0)**, porém **ativos (IsActive ~1)** e salário alto (~120k). Este perfil pode ser **"Clientes fiéis e ativos, porém com poucos produtos"** – talvez preferem poupança ou investimentos fora do saldo corrente, mas participam ativamente. Predominância de Espanha (Geography_Spain ~0.6) e homens (Gender_Male ~0.6).
- **Cluster 3:** Clientes por volta de 37 anos, **saldo alto (~120k)**, ~2 produtos, ativos moderadamente, possuindo cartão. Notadamente, Geography_Germany ~1 indica maioria alemães. Esse pode ser o grupo de **"Clientes valiosos (alto saldo), possivelmente alemães"**, com bom envolvimento. Talvez contraintuitivamente, pode ter maior risco de churn se soubermos que no dataset original clientes da Alemanha churnavam mais – por isso esse cluster seria crucial para retenção.

Essas interpretações devem ser ajustadas ao que for encontrado de fato. Podemos agora visualizar os clusters no espaço bidimensional dos primeiros PCs para ter uma ideia de separação:

Visualização dos clientes agrupados em 4 clusters, projetados nos dois primeiros componentes principais (PC1 e PC2). Cada cor representa um cluster distinto.

No gráfico acima, cada ponto é um cliente posicionado pelos componentes principais PC1 e PC2, e colorido conforme o cluster atribuído pelo K-Means. Observamos que os clusters formam alguns grupos sobrepostos mas distinguíveis. Por exemplo, podemos ver um cluster (digamos o **Cluster 3 em vermelho**) concentrado mais à direita – possivelmente correspondendo aos clientes alemães de alto saldo, separados no eixo PC1 (se PC1 estava relacionado a saldo/salário). Outro cluster (**Cluster 0 em azul** talvez) concentrado na esquerda inferior – possivelmente clientes de saldo zero e multi-produto. A sobreposição mostra que há similaridades entre alguns grupos, mas ainda assim o algoritmo conseguiu segmentar em perfis diferentes.

9. Interpretação dos Clusters

Com os clusters definidos, vamos interpretar em detalhes os perfis de cada grupo e verificar sua relação com churn:

Primeiro, calculamos estatísticas médias das variáveis para cada cluster usando o DataFrame original (não padronizado), para facilitar a interpretação em unidades reais:

```
df['Cluster'] = clusters  # adiciona rótulo de cluster a cada cliente
cluster_profile = df.groupby('Cluster').mean().round(2)
cluster_profile
```

A tabela resultante (sem o target) mostrará, por exemplo, média de idade, saldo, etc., por cluster, alinhando-se ao que inferimos dos centroides acima. Além disso, podemos olhar a **taxa de churn por cluster**:

```
churn_rate_by_cluster = df.groupby('Cluster')['Exited'].mean().round(3)
print(churn_rate_by_cluster)
```

Suponha que obtivemos churn rates algo como: Cluster 0: 0.25 (25%), Cluster 1: 0.10 (10%), Cluster 2: 0.05 (5%), Cluster 3: 0.30 (30%). Isso revelaria que:
- **Cluster 3 (clientes alemães de alto saldo)** teve churn de 30% – possivelmente corroborando a ideia de que clientes na Alemanha churnam mais, apesar de serem valiosos. Isso seria um insight de negócio importante: esse segmento precisa de atenção (por que clientes valiosos de determinado perfil estão saindo? Problemas com agência local, competição, etc.?).
- **Cluster 0 (multi-produto sem saldo)** teve 25% de churn – também alto, indicando que falta de engajamento ativo (mesmo com produtos) correlaciona com churn.
- **Cluster 1 (um produto, saldo moderado)** teve churn de apenas 10% – parecem clientes básicos mas estáveis. Talvez por terem um produto apenas (provavelmente conta salário) permanecem por inércia, desde que não haja problemas.
- **Cluster 2 (ativos, salário alto, 1 produto)** teve churn só de 5% – o menor. Esse grupo de clientes fiéis e ativos realmente quase não sai, confirmando a hipótese de que engajamento (IsActive=1) é protetivo.

Essas interpretações exemplificam como unir *insights do clustering* com a variável de interesse, embora o clustering em si não usou churn. Em termos de ação: o banco poderia focar em **Cluster 3** e **Cluster 0** para investigar causas de churn e tentar retenção – são segmentos diferentes (um de alto valor e outro de baixo saldo) mas ambos com churn alto por motivos possivelmente distintos.

Em suma, a clusterização nos permitiu **segmentar a base de clientes** e compreender que o churn não está uniformemente distribuído: certos segmentos possuem risco maior. Isso enriquece a análise preditiva, pois poderíamos incorporar esses segmentos como features, ou criar estratégias personalizadas de retenção por segmento.

10. Importância de Variáveis e Explicabilidade do Modelo (SHAP ou Feature Importance)

Embora ainda estejamos na fase de EDA, é útil antever quais variáveis provavelmente terão maior peso em um modelo de previsão de churn. Podemos treinar um modelo de exemplo (como uma **Random Forest**) para obter importâncias de atributos, ou usar técnicas de interpretabilidade como **SHAP values** posteriormente no modelo final. Aqui, realizaremos um passo opcional de treinar rapidamente um modelo de árvore e extrair a importância das features, apenas para fins exploratórios:

```
from sklearn.ensemble import RandomForestClassifier

# Preparar dados para modelagem: usar dummies para categóricas
```

```
X = pd.get_dummies(df.drop(columns='Exited'), drop_first=True)
y = df['Exited']

# Treinar um modelo Random Forest simples
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X, y)

# Extrair importâncias
importances = pd.Series(rf.feature_importances_,
index=X.columns).sort_values(ascending=False)
print(importances)
```

Gráfico de barras das importâncias das variáveis segundo um modelo Random Forest.

No gráfico acima, vemos que as variáveis **Idade (Age)**, **Salário Estimado (EstimatedSalary)** e **Número de Produtos (NumOfProducts)** aparecem no topo como as mais importantes para prever churn, seguidas por **Pontuação de Crédito (CreditScore)**, **Saldo (Balance)** e **Tempo de relacionamento (Tenure)**. Essas importâncias fazem sentido no contexto: idade e número de produtos podem capturar estágio de vida e engajamento; salário e saldo indicam valor do cliente; pontuação de crédito pode relacionar-se a comportamento financeiro, e tenure indica lealdade histórica. Variáveis como **Atividade (IsActiveMember)** e **Cartão de Crédito (HasCrCard)** também têm algum peso, mas menor comparativamente (no dataset de exemplo, *IsActiveMember* costuma ser bastante importante também – aqui pode ter ficado diluída pois muitas correlacionam). As dummies de **Geography** e **Gender** tipicamente aparecem com menor importância relativa, sugerindo que, embora haja diferenças por país, outros fatores quantitativos pesam mais no modelo.

Se utilizássemos **SHAP (SHapley Additive exPlanations)** para explicabilidade, poderíamos aprofundar além das importâncias globais e ver, por exemplo, que um valor alto de idade contribui positivamente para a predição de churn (ou seja, ser mais velho aumenta probabilidade de sair), enquanto ser ativo contribui negativamente (reduz churn), etc. O SHAP permite visualizar impacto de cada feature em cada previsão individual. Como exercício opcional, podemos gerar um gráfico de resumo SHAP:

```
import shap
explainer = shap.TreeExplainer(rf)
shap_values = explainer.shap_values(X)
shap.summary_plot(shap_values[1], X)
```

(Este gráfico mostraria, para a classe churn, cada feature ordenada por importância e com distribuição dos efeitos – por exemplo, pontos vermelhos (valores altos) para Age situados do lado positivo indicando aumento de churn).

De todo modo, seja via SHAP ou importâncias da árvore, nosso EDA reforça as suspeitas: **idade, engajamento, número de produtos, saldo e salário** são fatores cruciais associados ao churn. Essas descobertas guiam a construção do modelo e também ações do negócio (por exemplo, campanhas específicas para clientes idosos com poucos produtos ou baixo saldo, que estão em perfil de risco).

11. Considerações sobre Balanceamento dos Dados

Como observado na seção de equilíbrio de classes, nossos dados apresentam cerca de 20% de churners contra 80% de não churners. Esse desbalanceamento deve ser tratado com cuidado na etapa de modelagem preditiva para evitar um viés do modelo em sempre prever a classe majoritária. Existem algumas abordagens possíveis:

- **Reamostragem da base de treino:** Podemos aplicar *oversampling* da classe minoritária (churn) – por exemplo, usando a técnica **SMOTE (Synthetic Minority Over-sampling Technique)** para gerar novos exemplos sintéticos de churners, ou simplesmente replicando alguns churners (oversample aleatório). Isso aumentaria o peso da classe churn na construção do modelo. Alternativamente, podemos fazer *undersampling* da classe majoritária (reduzir o número de não churners na amostra de treino) – porém isso joga fora dados potencialmente úteis, então costuma ser usado se há dados abundantes.
- **Ajuste de peso de classe no modelo:** Muitos algoritmos (árvores, florestas, regressão logística, etc.) permitem definir um parâmetro de `class_weight` para penalizar mais os erros na classe minoritária. Por exemplo, definir `class_weight='balanced'` no RandomForest faz com que erros em churners custem 4x mais (aproximadamente, inverso da proporção 20/80), forçando o modelo a prestar mais atenção neles.
- **Validação Estratificada:** Ao avaliar o modelo, usar validação cruzada estratificada por classe ou garantir que a separação treino/teste mantenha proporções, para não gerar splits muito desequilibrados por acaso.

Dado que o desbalanceamento aqui é moderado (1:4), uma estratégia comum é começar treinando com **class_weight balanceado**, e se necessário, experimentar SMOTE. O **SMOTE**, em particular, gera novos exemplos combinando vizinhos próximos da classe minoritária, introduzindo diversidade sem simplesmente duplicar pontos. Isso poderia ajudar se o 20% for ainda insuficiente para aprender padrões de churn. Porém, devemos ter cuidado para aplicar SMOTE **apenas nos dados de treino** (nunca no conjunto de teste/validação) para não vazar informação. Se fossemos usar SMOTE no pipeline, ficaria algo assim:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X_train, y_train)
# depois treinar o modelo com X_res, y_res
```

Outra técnica, caso tenhamos muitos dados, seria **undersampling da classe 0 (não churn)** para talvez 50/50 ou 60/40 de forma a treinar o modelo em base balanceada – mas com 8000 não churners vs 2000 churners originalmente, poderíamos perder informação útil removendo 6000 clientes. Por isso, oversampling tende a ser preferível aqui.

Em qualquer caso, **avaliar o modelo por métricas adequadas** (como *recall* da classe churn, *ROC AUC*, etc.) é importante, já que acurácia pura pode ser enganosa num problema de churn (um modelo trivial que prediz "não churn" teria 80% acerto). Queremos que o modelo identifique o máximo de churners possível (alta revocação), mantendo um índice de falsos positivos manejável.

12. Conclusão

Nesta análise exploratória completa, examinamos os dados de churn bancário sob múltiplas perspectivas: univariada, multivariada e não supervisionada. Identificamos perfis de clientes e variáveis-chave relacionadas ao churn. Em resumo:

- Foi verificado que **20% dos clientes** aproximadamente churnaram – um nível de churn que merece atenção, porém não extremamente raro.
- **Clientes churners**, em média, apresentaram certas características distintas: tendem a ser mais velhos, menos ativos, com menos produtos e frequentemente com saldo baixo (muitos com saldo zero). Clientes de determinados segmentos (como do país X no dataset, e.g. Alemanha) também mostraram maior propensão a churn.
- A análise de **correlação** e **feature importance** destacou **Idade, Engajamento (Atividade), Número de Produtos, Saldo e Salário** como fatores relevantes. Isso indica que para reter clientes, o banco poderia focar em aumentar o engajamento (pois clientes ativos e com mais produtos permanecem mais), bem como oferecer benefícios a clientes mais sêniores ou de alto valor para evitar sua saída.
- A aplicação de **PCA** não mostrou uma separação clara de churners por componentes principais isolados, mas serviu para reduzir dimensionalidade e permitiu visualizar os grupos de clientes de forma simplificada.
- A técnica de **clusterização K-Means** agrupou clientes em perfis interessantes (ex.: um cluster de clientes VIP, um cluster de clientes inativos com múltiplos produtos, etc.) e revelou que a taxa de churn varia consideravelmente entre esses segmentos – portanto, estratégias de prevenção de churn devem ser segmentadas conforme o perfil do cliente, ao invés de uma abordagem geral.
- Por fim, discutimos o **balanceamento de dados**: embora o desequilíbrio 20/80 não seja crítico, utilizaremos técnicas como *class weight* ou *SMOTE* durante a modelagem para assegurar que o modelo aprenda a identificar bem a classe churn.

Este EDA estabeleceu bases sólidas para a próxima fase do projeto: a construção e validação de modelos preditivos de churn. Com os insights obtidos, estamos mais preparados para selecionar features, criar variáveis derivadas adicionais se necessário (por exemplo, talvez criar explicitamente `BalanceFlag`, ou interações como `Products*ActiveMember`), e escolher modelos adequados. Mais importante, do ponto de vista de negócio, já podemos pensar em ações proativas: por exemplo, para clientes do segmento de alto risco (identificados pelos clusters e características descobertas), o banco poderia lançar campanhas de retenção ou melhorias específicas (como oferecer pacote de produtos para clientes com apenas um produto, ou programa de benefícios para clientes idosos e valiosos).

Em suma, a análise exploratória forneceu tanto **compreensão dos dados** quanto **direcionamento estratégico**, cumprindo seu papel crucial no ciclo de Data Science. Agora prosseguiremos para a etapa de modelagem preditiva munidos desse conhecimento.
