ChatGPT

Projeto de Previsão de Churn Bancário – Banco Montes Claros

Objetivo do Projeto

O **Banco Montes Claros** (instituição fictícia) identificou um problema de negócios crítico: a evasão de clientes (churn) nos serviços de cartão de crédito. Reter clientes existentes é muito mais rentável do que adquirir novos – estudos indicam que conquistar um novo cliente pode custar 5 a 25 vezes mais do que manter um atual. Diante disso, o projeto teve **como objetivo geral desenvolver um modelo preditivo de churn** que identificasse quais clientes estão propensos a abandonar o banco, permitindo **ações proativas de retenção**. Em suma, buscou-se criar uma solução de *Machine Learning* que não apenas obtivesse boa acurácia, mas também gerasse **insights acionáveis** para aumentar a retenção e maximizar o valor dos clientes para o banco.

Objetivos específicos incluíram:
- **Análise Exploratória de Dados (EDA)** para entender o perfil dos clientes e identificar fatores que influenciam a saída (churn).
- **Modelagem preditiva supervisionada (classificação)**: construir e comparar vários modelos de *Machine Learning* (Regressão Logística, Árvore de Decisão, Random Forest, XGBoost, LightGBM, etc.) para prever churn.
- **Segmentação de clientes** via análise de **cluster** *não supervisionada* (K-Means), visando agrupar clientes com características similares e possibilitar estratégias diferenciadas por segmento.
- **Estimativa de Lifetime Value (LTV)**: calcular um proxy de LTV para cada cliente/segmento, estimando o valor de longo prazo de cada cliente e integrando essa métrica com o risco de churn.
- **Cenários de negócio e recomendações**: usando as previsões de churn e LTV, simular cenários (por exemplo, impacto financeiro se nada for feito vs. se ações de retenção forem tomadas) e propor ações estratégicas ao banco.
- **Implementação**: Desenvolver uma aplicação web interativa (usando Streamlit) para demonstrar o modelo em funcionamento e permitir a exploração dos resultados por usuários finais.

Conjunto de Dados e Análise Exploratória 📊

O dataset utilizado foi **BankChurners.csv**, contendo informações de **10.127 clientes** de cartão de crédito, dos quais **1.627 são churners** (indicados pela coluna *Attrition_Flag* = "Attrited Customer"), o que representa uma taxa de churn de aproximadamente **16%** [1]. Cada registro corresponde a um cliente, com variáveis demográficas (idade, gênero, faixa de renda, nível de educação), financeiras (limite de crédito, saldo rotativo, etc.) e comportamentais (número e valor de transações, meses de inatividade, contagem de contatos de atendimento, entre outras) [1] [2]. Por exemplo, o dataset inclui campos como **Customer_Age**, **Gender**, **Income_Category**, **Credit_Limit**, **Total_Trans_Ct** (número total de transações nos últimos 12 meses), **Total_Trans_Amt** (valor total gasto), **Months_Inactive_12_mon** (meses inativo no último ano), **Contacts_Count_12_mon** (número de contatos/SAC no último ano), além de variáveis que medem mudança no uso (como **Total_Amt_Chng_Q4_Q1** e **Total_Ct_Chng_Q4_Q1**, variação de gastos e transações do 4º tri para o 1º tri) e a variável alvo **Attrition_Flag** indicando se o cliente deixou o banco [2].

A análise exploratória inicial confirmou o **desbalanceamento de classes** significativo: cerca de ~84% dos registros são de clientes ativos vs. ~16% churners [1]. Esse desequilíbrio implica que, sem tratamento, um modelo trivial que previsse "não churn" para todos já obteria ~84% de acurácia, o que **reforça a necessidade de técnicas de balanceamento e métricas adequadas** focadas na classe minoritária (churn) [3]. Foram gerados gráficos da distribuição das classes, evidenciando essa disparidade e motivando abordagens para lidar com o desbalanceamento dos dados [4] [5].

Durante a EDA, explorou-se a distribuição e relação das variáveis com o churn. Descobriu-se **padrões claros nos drivers de churn**: clientes que apresentam **maior engajamento tendem a não churnar**, enquanto sinais de insatisfação ou baixo uso correlacionam-se positivamente com churn. Por exemplo, notou-se que clientes com **muitos meses de inatividade** e **alto número de contatos de atendimento (SAC)** têm **probabilidade drasticamente maior de churn**, sugerindo desengajamento e possíveis problemas de satisfação. Em contrapartida, clientes com **uso intenso do cartão (muitos gastos/transações)** e **relacionamento amplo com o banco (vários produtos)** mostraram propensão menor a sair [6]. Essas observações iniciais – como "**mêses inativos > 2**" ou "**contatos de SAC elevados**" indicando risco, versus uso ativo e multiplos produtos indicando fidelidade – alinharam-se bem com a intuição de negócio e posteriormente seriam confirmadas pelos modelos e pela análise de importância de variáveis.

Além disso, analisaram-se correlações entre variáveis para evitar multicolinearidade e entender redundâncias. Identificou-se, por exemplo, que variáveis derivadas como **Total_Trans_Amt** (gasto anual) e **Total_Trans_Ct** (freqüência de transações) são correlacionadas, mas ambas agregam valor preditivo. A EDA também permitiu verificar que não havia dados faltantes significativos no dataset original, embora algumas variáveis categóricas precisassem de tratamento (ex.: *Income_Category* possui faixas como "Less than 40K", "40-60K", etc., que foram ordenadas ou convertidas em classes numéricas posteriormente).

Preparação dos Dados e Engenharia de Atributos 🏗️

Antes da modelagem, realizou-se um extenso pré-processamento para limpar e enriquecer os dados, resultando em uma base tratada (`base_tratada.csv`) pronta para os algoritmos. As etapas de preparação incluíram:

- **Limpeza e seleção de atributos**: Remoção de colunas irrelevantes para previsão, como identificadores únicos (*Client ID*) que não agregam valor preditivo. Por exemplo, a coluna `CLIENTNUM` (ID do cliente) foi descartada por não contribuir na modelagem. Também foram padronizados nomes de variáveis e verificada a consistência dos tipos de dados (conversão de campos numéricos/categóricos conforme apropriado).

- **Criação de Variáveis Derivadas**: Uma importante contribuição do projeto foi a **engenharia de atributos** (*feature engineering*), criando variáveis que capturam melhor o comportamento e valor dos clientes. Entre as principais variáveis derivadas adicionadas estão [7]:

- `Ticket_Medio` – valor médio por transação (indicador de ticket médio de gastos). *Exemplo de cálculo:* se um cliente gastou `Total_Trans_Amt = 5000` em `Total_Trans_Ct = 100` transações, seu ticket médio será `5000/100 = 50` [7]. `python`
  `# Exemplo de cálculo de Ticket Médio por transação`
  `df['Ticket_Medio'] = df['Total_Trans_Amt'] / df['Total_Trans_Ct']`

- `Gasto_Medio_Mensal` – gasto médio mensal no cartão. Como o dataset abrange ~12 meses de uso, essa feature aproxima o quanto, em média, o cliente gasta por mês (por exemplo, dividindo *Total_Trans_Amt* por 12) [7].
- `Rotativo_Ratio` – razão de uso de crédito rotativo, relacionando o saldo rotativo (*Total_Revolving_Bal*) com o limite de crédito (*Credit_Limit*). Esse índice indica o quanto do limite o cliente costuma deixar pendente (próximo do *Avg_Utilization_Ratio* do dataset original) [7].
- `Score_Relacionamento` – uma pontuação agregada de relacionamento do cliente com o banco. Por exemplo, combinando o número de produtos que possui (*Total_Relationship_Count*) com o tempo de casa (*Months_on_book*) e possivelmente outros fatores de engajamento, para refletir quão "envolvido" o cliente está.
- `Caiu_Valor` e `Caiu_Transacoes` – variáveis binárias (flags) indicando **queda de uso**: sinalizam se houve redução no valor gasto ou no número de transações em comparação ao período anterior. Baseiam-se nas variáveis de mudança trimestral (Q4 vs Q1): por exemplo, se *Total_Amt_Chng_Q4_Q1* < 1 (ou seja, gasto no último trimestre caiu em relação ao primeiro trimestre), `Caiu_Valor = 1`; similarmente, *Total_Ct_Chng_Q4_Q1* < 1 indicaria `Caiu_Transacoes = 1`. Essas flags capturam tendência de queda no uso do cartão, potencialmente um alerta de churn [7].
- `LTV_Proxy` – estimativa proxy do **Lifetime Value** do cliente (discutida detalhadamente adiante). Resume o valor financeiro de longo prazo do cliente para o banco, combinando seu volume de gastos, uso de crédito rotativo e relacionamento em uma única métrica [7].
- `Faixa_Idade` – categoria de idade (e.g., 20-30, 30-40, etc.) para incorporar possíveis efeitos não lineares ou segmentações etárias no comportamento de churn.
- `Renda_Class` – categorização da renda (derivada de *Income_Category*) ordenada de menor para maior renda, ou agrupada em classes como "Baixa, Média, Alta".

Obs: Todas as novas variáveis foram documentadas em um dicionário de dados anexo e validadas quanto à consistência durante a EDA. Essas features derivadas trouxeram informações adicionais além das variáveis originais, potencialmente melhorando o poder preditivo do modelo [7].

- **Transformação de Variáveis**: Aplicaram-se transformações adequadas nos dados para prepará-los aos algoritmos de ML:
- *Encoding de categóricas:* Variáveis categóricas como **Gender**, **Education_Level**, **Marital_Status**, **Income_Category** e **Card_Category** foram codificadas em formato numérico. Utilizou-se técnicas como **one-hot encoding** (via *pandas get_dummies* ou *OneHotEncoder* do scikit-learn) para representar categorias sem pressupor ordenação, resultando em colunas binárias (0/1) para cada categoria. Isso aumentou a dimensionalidade dos dados, mas era necessário para incluir essas informações nos modelos baseados em algoritmos lineares ou de árvore.
- *Normalização/Escalonamento:* As variáveis numéricas contínuas foram **padronizadas** (*standardization*) para terem média 0 e desvio padrão 1 antes de certos algoritmos. Em especial, para métodos baseados em distância (ex.: K-Means, PCA) e para regressão logística, aplicou-se o *StandardScaler*. Isso garante que diferenças de escala (por exemplo, valores monetários vs. contagens) não distorçam os resultados [8]. *(Nota:* Algoritmos baseados em árvore como Random Forest e XGBoost não necessitam de normalização, mas ainda assim foi mantida a consistência de escalonamento dentro de *pipelines* de preprocessamento para o modelo final quando apropriado).
- *Tratamento de outliers:* Identificaram-se outliers em variáveis como **Total_Trans_Amt** (alguns clientes com gastos excepcionalmente altos, p.ex. ~\$18k anuais) [9]. Para mitigar seu impacto sem removê-los, utilizou-se *winsorization* (cap) nos valores extremos – por exemplo, limitando valores acima do percentil 99 ao valor desse percentil [9]. Dessa forma, reduz-se a influência desproporcional de pontos muito fora da curva, especialmente importante para a clusterização e modelos sensíveis.

- *Divisão Treino/Teste:* O dataset foi dividido em subsets de treinamento e teste, reservando aproximadamente **30% dos dados para teste** final do modelo (proporção estratificada, preservando a taxa de churn na divisão). A divisão estratificada garante que a distribuição de churn no conjunto de teste (~16%) reflita a realidade, permitindo avaliar a performance do modelo de forma honesta.
- *Pipelines e reprodutibilidade:* Todas as transformações aplicadas nos dados de treino (encoding, escalonamento, criação de variáveis) foram encapsuladas em pipelines utilizando *scikit-learn*, de forma que pudessem ser reaplicadas **exatamente** da mesma maneira nos dados de teste e futuros. Os objetos de transformação (ex.: encoders, scalers) foram **salvos** após o treino e recarregados para transformar o conjunto de teste, garantindo consistência e evitando qualquer vazamento de informação do futuro [10].

Após essas etapas, chegou-se a uma base tratada enriquecida (`base_tratada.csv`), pronta para a modelagem. Todo o código de preparação de dados foi organizado tanto em notebooks (para análise passo-a-passo) quanto em scripts automatizáveis (por ex., `src/preparacao_dados.py` pode gerar o dataset limpo via linha de comando). Isso permitiu refazer o processamento facilmente e documentou o pipeline de dados para eventuais atualizações.

Balanceamento de Classes ⚖

Devido ao desbalanceamento (apenas ~16% de churners), foi imprescindível adotar estratégias para que os modelos **não ficassem tendenciosos para a classe majoritária** (não-churn). Sem correções, um modelo poderia ter alta acurácia simplesmente prevendo *"cliente não sai"* para todos, mas seria inútil para detectar churn. Assim, foram empregadas várias técnicas de **balanceamento de classes** durante o desenvolvimento dos modelos:

- **Métricas apropriadas:** Desde o início, ficou claro que métricas como **Recall** (sensibilidade para classe churn) e **AUC-ROC** seriam mais indicativas de performance do que a acurácia bruta [11]. Avaliar os modelos considerando **F1-Score**, **Precision** e principalmente **Recall do churn** garantiu foco na capacidade de identificar churners, mesmo que às custas de alguns falsos positivos [12]. Esse cuidado metrificou melhor o sucesso dado o objetivo de reter clientes (melhor errar marcando alguém como risco do que deixar churners passarem despercebidos).

- **Rebalanceamento via pesos de classe:** Para muitos algoritmos, utilizou-se o parâmetro de **peso de classe balanceado** fornecido por bibliotecas como scikit-learn. Por exemplo, na Regressão Logística e na Random Forest, ajustou-se `class_weight='balanced'`, de forma que erros em prever um churn (classe minoritária) fossem **penalizados mais fortemente** do que erros na classe majoritária [13]. Na prática, isso faz o modelo "dar mais atenção" aos churners durante o treino, corrigindo o viés natural.

- **Oversampling e undersampling (testes):** Experimentos foram conduzidos gerando conjuntos de treino balanceados através de técnicas de **oversampling** (aumento sintético da classe minoritária) e **undersampling** (redução da majoritária). Especificamente, avaliou-se o uso de **SMOTE** (*Synthetic Minority Over-sampling Technique*) para criar exemplos artificiais de churners, bem como duplicação simples de churners existentes, e o contraponto de undersampling de não-churners [14]. Também foi testada a combinação das duas abordagens (balanceamento híbrido). Esses conjuntos balanceados alimentaram alguns modelos em estágios de experimentação para verificar se melhorariam a detecção de churn.

Resultado: Os melhores resultados foram obtidos **sem a necessidade de dados sintéticos**, mas sim usando **pesos de classe e modelos robustos**. A técnica de *class weight* mostrou-se suficiente e mais controlada – os modelos (especialmente ensembles) aprenderam a focar nos churners sem precisar de oversampling exagerado [15]. Em alguns casos, o SMOTE e oversampling até melhoraram ligeiramente o recall, mas ao custo de mais falsos positivos. Optou-se, portanto, por **não incorporar exemplos sintéticos no modelo final**, privilegiando o ajuste de peso e algoritmos ensembles que por natureza lidam melhor com classes desbalanceadas [15]. Vale destacar que, paralelamente, utilizou-se **validação cruzada estratificada** nas etapas de treinamento/validação para garantir que a proporção de churners fosse mantida em cada *fold*, dando confiança de que a performance medida generalizaria.

Modelagem Supervisionada – Algoritmos Testados

Com os dados prontos, partiu-se para a construção de modelos de classificação para prever churn (Attrition_Flag). Adotou-se uma abordagem pragmática de **testar múltiplas técnicas** de *Machine Learning* e comparar seus desempenhos, seguindo o escopo do projeto. No total, foram treinados e avaliados os seguintes modelos principais:

- **Regressão Logística:** Serviu como **baseline** inicial por sua simplicidade e interpretabilidade. Aplicou-se regularização (testando penalização L1 *Lasso* vs L2 *Ridge*) e calibraram-se hiperparâmetros como o coeficiente de regularização (*C*) via busca em grade. Também testou-se a versão padrão vs. `class_weight='balanced'` do scikit-learn para lidar com o desbalanceamento [16]. A regressão logística apresentou desempenho razoável e, apesar de não ser a mais acurada, seus coeficientes forneceram insights valiosos: por exemplo, coeficientes positivos fortes para **Months_Inactive_12_mon** e **Contacts_Count_12_mon** confirmaram que muitos meses inativo e muitos contatos de SAC aumentam a chance de churn – coerente com a análise exploratória [17]. Esse modelo baseline interpretável ajudou a validar a importância das variáveis e serviu de comparação para modelos mais complexos.

- **Árvore de Decisão:** Uma *Decision Tree* foi treinada com profundidade máxima controlada e mínimo de amostras por folha, para evitar overfitting [18]. A árvore isolada capturou algumas regras de decisão compreensíveis – por exemplo: *"IF Months_Inactive_12_mon > 2 e Total_Trans_Ct é baixo THEN churn"*, refletindo exatamente o padrão esperado (clientes pouco ativos e pouco movimentados tendem a sair) [19]. Apesar de fácil interpretação, a árvore única teve performance limitada (tendência a superajustar à maioria, como veremos nas métricas), indicando que seria necessário um modelo mais robusto.

- **Random Forest:** Um ensemble de múltiplas árvores foi então treinado. Usou-se uma **Random Forest** com ~100 árvores (*n_estimators* $\approx$ *100*), ajustando parâmetros como *max_depth* e *max_features* via validação cruzada e *Grid Search* [20]. O Random Forest trouxe melhoria significativa em relação aos modelos anteriores, devido à combinação de árvores reduzir overfitting e capturar mais variabilidade. Notavelmente, ao inspecionar as **feature importances** deste modelo, destacaram-se variáveis como **Total_Trans_Ct**, **Months_Inactive_12_mon**, **Contacts_Count_12_mon**, **Total_Relationship_Count** e **Total_Trans_Amt** entre as mais importantes [21]. Isso corroborou os achados da EDA, indicando que volume de transações, engajamento (ou falta dele) e relacionamento amplo são realmente os principais drivers de churn. Com boa acurácia e equilíbrio entre precisão/recall, a Random Forest já se mostrou um candidato forte.

- **XGBoost (Extreme Gradient Boosting):** Este modelo de boosting gradient foi testado dado seu histórico de alto desempenho em problemas de classificação tabular. Realizou-se tuning de

hiperparâmetros como o número de árvores (*n_estimators*), taxa de aprendizado (*learning_rate*), profundidade máxima das árvores, e parâmetros de regularização (ex.: *gamma*, *subsample*, *colsample_bytree*), principalmente via *Randomized Search* com validação cruzada [22]. Também aplicou-se **early stopping** durante o treino para prevenir overfitting (parando se a métrica de validação não melhorasse após ~10 rounds). O **XGBoost obteve um dos melhores desempenhos**, conseguindo elevar tanto a acurácia quanto, especialmente, o **recall de churn**. Por conseguir capturar relacionamentos não lineares complexos entre variáveis, o modelo detectou padrões sutis e apresentou robustez mesmo com dados desbalanceados.

- **LightGBM:** Testou-se em paralelo o **LightGBM** (implementação de boosting da Microsoft), de forma similar ao XGBoost. Ajustaram-se seus parâmetros (como *num_leaves*, *learning_rate*, *feature_fraction*, etc.) e obteve-se performance **praticamente equivalente à do XGBoost** em nossa validação [23]. O LightGBM por vezes teve AUC ligeiramente maior ou menor (<0.002 de diferença), dentro da margem. Embora muito próximos, o XGBoost apresentou uma sensibilidade ao churn marginalmente superior em alguns folds, razão pela qual optamos por ele como modelo final principal [23]. Ainda assim, o LightGBM foi **salvo como modelo de comparação**, pois possui a vantagem de ser mais rápido e leve em produção – podendo ser útil considerar uma versão otimizada dele dependendo do caso de uso futuro [24].

(Também foram considerados brevemente algoritmos como K-Nearest Neighbors (KNN) e SVM. Porém, devido ao tamanho relativamente grande do dataset (10k) e à alta dimensionalidade após o one-hot encoding*, esses métodos se mostraram menos viáveis – KNN seria computacionalmente custoso e o SVM teve dificuldade em convergir – por isso não foram priorizados nos resultados finais [25].)*

Validação e Seleção de Modelos: Para cada modelo acima, aplicou-se **validação cruzada estratificada (5-fold)** no conjunto de treino, juntamente com *grid search* ou *random search* de hiperparâmetros conforme citado, para encontrar configurações ótimas [26]. As métricas de validação orientaram a seleção do modelo final. Observou-se consistentemente os ensembles (Random Forest, XGBoost, LightGBM) superando os modelos mais simples em métricas-chave. A **tunagem de hiperparâmetros** elevou o desempenho – por exemplo, reduzir o *learning_rate* e aumentar *num_leaves* no LightGBM melhorou a AUC em ~5 pontos percentuais [27] [28], além de equilibrar melhor precision/recall. Ao final, definiu-se o XGBoost como modelo preferido, com LightGBM em segundo muito próximo, e Random Forest logo em seguida.

Avaliação de Desempenho 📈

Após escolher os hiperparâmetros finais, procedeu-se à **avaliação no conjunto de teste reservado (30% dos dados)**, nunca usado no treino, para obter uma medida realista de performance. As principais **métricas consideradas** foram: **Acurácia**, **ROC AUC**, **F1-Score**, **Precisão (Precision)** e **Recall (Sensibilidade)** focados na classe churn, além da análise da **Matriz de Confusão**. Como discutido, a acurácia sozinha poderia enganar devido à classe desbalanceada, então o foco esteve em métricas como AUC e, principalmente, recall do churn (proporção de churners corretamente identificados) e precision (proporção de predições de churn que realmente eram churn).

Os resultados obtidos no teste (aproximados) foram:

- **Regressão Logística:** Acurácia ~85%, **Recall ~78%**, Precision ~56% (F1 ~65%). A matriz de confusão mostrou que o modelo conseguiu detectar cerca de **78% dos churners** (sensibilidade 0.78), porém com alguns falsos positivos (precision 0.56) [29]. A AUC ficou em torno de 0.84, indicando performance moderada [29].

- **Árvore de Decisão:** Acurácia ~90%, porém **Recall ~61%** apenas. O modelo overfitou um pouco a classe majoritária – priorizando acertar "não churn" – resultando em recall baixo para churn (sensibilidade 0.61) [30]. A AUC ~0.80 confirma a performance inferior. Devido a esse **desbalanceamento de erros (muitos churners perdidos)**, a árvore isolada não foi utilizada isoladamente na solução final [30].

- **Random Forest:** Acurácia ~93%, **Recall ~68%**, Precision ~85% (F1 ~75%). Houve clara melhoria: o modelo capturou mais churners (68%) mantendo boa precisão (85%), indicando poucos falsos alarmes [31]. A **ROC AUC ~0.94** demonstrou excelente separabilidade entre classes [31]. A *Balanced Accuracy* situou-se em ~0.83, mostrando bom equilíbrio. Em resumo, o Random Forest conseguiu elevar significativamente o recall em comparação à árvore simples, sem sacrificar muito a precisão.

- **XGBoost:** Acurácia ~92%, **Recall ~82%**, Precision ~71% (F1 ~76%). Este modelo **se destacou** por identificar **mais de 80% dos clientes churn** (sensibilidade ~0.82), mantendo uma precisão aceitável ~71% [32]. A *Balanced Accuracy* chegou perto de 0.88 e a AUC ~0.96 [32], a mais alta entre os modelos testados. Ou seja, o XGBoost foi **mais agressivo em pegar churners**, conseguindo achar 4 em cada 5 que realmente churnaram, ao mesmo tempo em que manteve uma taxa de falsos alarmes relativamente controlada. Dada a prioridade do negócio em não perder quem vai sair, este modelo apresentou o melhor *trade-off* para nosso objetivo [32] [33].

- **LightGBM:** Acurácia ~91%, **Recall ~80%**, Precision ~72% (F1 ~76%). Praticamente **equivalente ao XGBoost** dentro da margem de erro [34]. O LightGBM alcançou ~80% de recall e ~72% de precisão, com AUC também ~0.96. Essas diferenças mínimas confirmaram que ambos os boosters performam muito bem no dataset, com ligeira vantagem do XGBoost em recall.

Observações: A **acurácia geral alta (>90%)** em vários modelos deve-se em parte ao grande número de negativos (não-churn) que a maioria dos modelos acerta facilmente [35]. Por isso, enfatizamos **Recall do churn e AUC** como medidas de sucesso. O modelo de destaque (XGBoost) atingiu ~82% de recall no churn, ou seja, **identificou 4 de cada 5 clientes que saíram**, algo que representa uma melhoria enorme frente ao baseline trivial (que pegaria 0 de 5) [33]. Ao mesmo tempo, a precisão ~71% significou que cerca de 29% dos alertas seriam falsos – um patamar considerado administrável para fins de retenção proativa (é melhor alertar em excesso do que deixar churners escaparem).

Para visualizar a performance, foram incluídos no relatório gráficos e tabelas: a **matriz de confusão** do modelo final (mostrando verdadeiros/falsos positivos e negativos), a **curva ROC** comparativa dos principais modelos, e uma tabela resumindo as métricas (Acurácia, Recall, Precision, F1, AUC) de cada modelo [36]. Esses artefatos ajudaram a comunicar aos stakeholders os ganhos obtidos – por exemplo, mostrando que o XGBoost elevou o recall de churn de ~60% (árvore) para ~82%, com AUC próxima de 0.96.

Conforme esperado, os métodos baseados em ensemble (**Random Forest, XGBoost, LightGBM**) superaram a regressão logística e a árvore simples [37]. No entanto, a **Regressão Logística** provou seu valor em **interpretabilidade** – seus coeficientes permitiram explicações claras sobre o efeito de cada variável no churn, alinhadas ao conhecimento de negócio. Como bem pontuado, *"análises de regressão ajudam a entender a relação entre atributos e churn do cliente, oferecendo insights acionáveis para estratégias de retenção"* [38]. Assim, combinamos o melhor dos dois mundos: modelos avançados para performance e modelos simples para insights.

Modelo Final Selecionado

Ao término da competição de modelos, escolhemos o **XGBoost como modelo preditivo final de churn** para implementação. A decisão considerou seu excelente desempenho (AUC ~0.96, recall 82% no teste) aliado a tempos de predição rápidos e escalabilidade. O modelo final foi treinado em todo o conjunto de treinamento otimizado e então **salvo em disco** usando a biblioteca *pickle* (`pickle.dump`). O arquivo resultante `modelo_churn_xgb.pkl` armazena o modelo treinado e foi incluído nos artefatos do projeto [39]. Este será o modelo carregado pela aplicação web para gerar previsões.

(Nota: O LightGBM otimizado também foi salvo como `modelo_churn_lgbm.pkl` para referência futura, já que tem praticamente a mesma acurácia e poderia ser útil em um cenário onde desempenho computacional for crítico. Porém, para efeitos do produto final e da apresentação, utilizamos o XGBoost como o preditor principal.)

Importância de Atributos e Interpretação do Modelo 🔍

Com um modelo de alta performance em mãos, foi dedicado esforço para torná-lo **interpretável** e extraír insights dos **fatores que influenciam o churn**. Várias abordagens foram utilizadas para interpretar o modelo e validar se ele estava alinhado ao domínio do negócio:

- **Coeficientes da Regressão Logística:** Como mencionado, a análise dos coeficientes do modelo logístico (mesmo não sendo o modelo final) serviu para **confirmar a influência das variáveis** de forma transparente. Viu-se claramente, por exemplo, que **cada mês adicional de inatividade** eleva significativamente a chance prevista de churn (coeficiente positivo alto para *Months_Inactive_12_mon*), e que **mais contatos ao SAC** também contribuem para churn (coef. positivo para *Contacts_Count_12_mon*) [17]. Por outro lado, variáveis como **Total_Trans_Ct** apresentaram coeficiente negativo na regressão, indicando que mais transações reduzem a probabilidade de churn – tudo consistente com nosso entendimento do problema.

- **Feature Importance Global (árvores):** Os modelos de árvore forneceram métricas de importância de atributos. Em especial, foi gerado um gráfico de **Feature Importance** do **Random Forest** e do **XGBoost**, ordenando as variáveis por importância relativa [40]. As features no topo destes gráficos foram exatamente aquelas esperadas: *Total_Trans_Ct*, *Months_Inactive_12_mon*, *Contacts_Count_12_mon*, *Total_Relationship_Count*, *Total_Trans_Amt*, etc. [21] [41]. Essa análise confirmou quantitativamente os principais drivers de churn identificados na EDA. Por exemplo, o XGBoost atribuiu grande ganho às variáveis de (falta de) engajamento e uso do cartão, reforçando que o modelo aprendeu os padrões corretos.

- **Interpretação SHAP:** Para aprofundar a interpretabilidade do modelo final (uma vez que mesmo feature importances globais de boosters ainda são relativamente *caixa-preta*), recorreu-se à técnica de **SHAP (SHapley Additive exPlanations)**. No notebook de interpretabilidade (`Feature_Importance_SHAP.ipynb`), calculamos os valores SHAP do modelo XGBoost em um conjunto de dados de exemplo (como o conjunto de teste) e geramos visualizações para explicar as previsões. Em especial:

- **SHAP Summary Plot:** um gráfico do tipo *beeswarm* que mostra a distribuição dos valores SHAP de cada feature, ordenadas por importância média [42]. Esse gráfico permitiu ver não apenas quais variáveis mais contribuem para o modelo, mas também **em que direção** o fazem – por exemplo, pontos vermelhos (valores altos da variável) posicionados à direita (SHAP positivo) indicam que valores altos daquela feature aumentam a probabilidade de churn. Observou-se

nesse plot que clientes com **muitos meses inativos** tendem a ter SHAP positivo elevado (empurrando a predição para churn), enquanto clientes com **muitas transações** têm SHAP negativo (reduzindo o churn previsto), alinhando perfeitamente com as expectativas.

- **Gráfico de Dependência:** foram gerados *SHAP dependence plots* para 2-3 variáveis principais, como *Months_Inactive_12_mon* e *Total_Trans_Ct*. Esses gráficos mostraram a relação entre o valor da feature e seu impacto SHAP no output (eixo y), revelando tendências – por exemplo: *à medida que o número de transações cai para valores baixos, o impacto no churn (SHAP) torna-se fortemente positivo*, indicando que poucos transações de fato contribuem muito para churn (enquanto altas transações têm impacto negativo) [43] [44].
- **Explicações Locais:** utilizamos plots específicos (como *force plot* e *waterfall plot*) para **explicar predições individuais**. Selecionando alguns clientes churn e não-churn, decompusemos a previsão do modelo na soma das contribuições de cada atributo. Por exemplo, para um determinado cliente com alta probabilidade de churn (80%), o gráfico waterfall mostrou setas vermelhas indicando que *Months_Inactive_12_mon = 5* e *Contacts_Count_12_mon = 6* impulsionaram a previsão para cima, enquanto uma *Total_Trans_Ct* baixa contribuiu mais um pouco – resultando na alta propensão final. Essas visualizações ajudaram a entender caso a caso **por que o modelo deu aquela previsão**, aumentando a confiança de que ele utiliza padrões corretos e identificando o que "explicar" ao gerente sobre um cliente em risco.

Em resumo, as técnicas de interpretabilidade (coeficientes, importâncias e SHAP) indicaram consistentemente que o modelo captura fatores intuitivos: **engajamento baixo e insatisfação levam ao churn**, enquanto **uso ativo e relacionamento amplo retêm clientes**. Isso ofereceu **transparência** na solução, crucial para a adoção prática. Ainda sugeriu que o modelo **não estava apoiando suas decisões em variáveis indevidas** (por exemplo, *Income* ou *Gender* não apareceram entre os tops, nem tiveram SHAP significativo), dando tranquilidade quanto a vieses.

(Cabe pontuar que, devido ao escopo, a implementação de SHAP foi utilizada principalmente como análise de apoio no notebook e para geração de gráficos ilustrativos anexos. Para um deploy em produção, poder-se-ia incorporar SHAP ou LIME diretamente no app para que cada previsão venha acompanhada de uma explicação – isso foi listado como possível melhoria futura [45] [46].)

Clusterização de Clientes – Segmentação

Além da predição de churn em si, o projeto requereu uma **análise não supervisionada** para segmentar a base de clientes do Banco Montes Claros em grupos com comportamentos similares. O objetivo foi identificar **clusters de clientes** que pudessem ser alvos de estratégias diferenciadas – por exemplo, descobrir um grupo de clientes **"VIP fidelizados"** vs **"insatisfeitos de alto risco"** – enriquecendo a compreensão do churn e permitindo intervenções mais focadas [47].

Preparação para Clustering

Para a clusterização, **focamos nos clientes ativos (não churnados)**, pois faz mais sentido segmentar aqueles que ainda estão na base para ações futuras [48]. Os clientes que já churnaram foram deixados de fora na formação dos grupos, embora posteriormente analisássemos **em qual cluster os churners se enquadravam** – isso para verificar se certos clusters tinham maior propensão a churn, retroalimentando insights [48].

Selecionamos um conjunto de **variáveis relevantes** para definir os clusters, priorizando atributos de perfil de uso e valor, em vez de demográficos puros. Entre as variáveis usadas estavam [49]:

- **Comportamento de uso:** ex. **Total_Trans_Amt**, **Total_Trans_Ct** (nível de atividade no cartão, volume e frequência de transações), **Months_Inactive_12_mon**, **Contacts_Count_12_mon** (indicadores de *disengagement* ou atrito), **Avg_Utilization_Ratio** (percentual do limite usado, ligado a uso de crédito rotativo) [49]. Essas variáveis capturam quão ativamente o cliente utiliza o cartão e quão engajado ou insatisfeito ele pode estar.
- **Relacionamento com banco: Months_on_book** (tempo de relacionamento, antiguidade), **Total_Relationship_Count** (número de produtos que o cliente possui com o banco) [50]. Indicam profundidade do relacionamento.
- **Demografia simplificada:** incluiu-se de forma **limitada** variáveis demográficas – evitamos peso excessivo de idade, renda, etc., para que os clusters se diferenciassem principalmente por comportamento. Testou-se incluir a **idade** ou uma indicação de **renda** (ex.: alta vs baixa), mas de forma controlada [51]. Queríamos que os clusters fossem mais *acionáveis* em termos de estratégias (focadas em comportamento e valor, que o banco pode influenciar), do que segmentações puramente demográficas.
- Importante: **A variável de churn NÃO foi usada** no algoritmo de clustering (K-Means é não supervisionado) [52]. Ela serviu apenas depois, para rotular/interpretar os clusters ("cluster X tem alta taxa de churn?"), mas não para formá-los.

Antes de rodar o algoritmo, **padronizamos todas as variáveis numéricas** (média=0, desvio=1) para garantir que diferenças de escala não distorcessem as distâncias calculadas pelo K-Means [8]. Aplicamos *StandardScaler* nas variáveis selecionadas. Também lidamos com outliers conforme citado – por exemplo, valores extremos de **Total_Trans_Amt** foram capados no percentil 99 – para que um ou dois clientes fora da curva não criassem um cluster só deles indevidamente [9]. Com os dados assim normalizados e tratados, prosseguimos à clusterização.

Definição do Número de Clusters (K)

Optamos pelo algoritmo **K-Means** pela sua simplicidade e interpretabilidade inicial. Porém, era necessário definir o número de clusters *K*. Não havendo um valor trivial de antemão, utilizamos métodos de avaliação para escolher um K adequado:

- **Método do Cotovelo (Elbow Method):** Executamos o K-Means para diversos valores de K (de 2 a 10 clusters, por exemplo) e calculamos a soma das distâncias quadráticas intra-cluster (**inertia**) para cada K. Plotamos a inertia vs K e procuramos o "cotovelo" na curva – ponto onde adicionar mais clusters passa a trazer ganho marginal pequeno. Observamos que a curva apresentava queda acentuada até por volta de **K = 3 ou 4**, estabilizando depois [53]. Isso sugeriu que **3 ou 4 clusters** capturariam a maior parte da heterogeneidade, e além disso os ganhos eram decrescentes.

- **Coeficiente de Silhueta:** Complementarmente, calculamos o **Silhouette Score** médio para diferentes K. Esse coeficiente varia de -1 a 1 e mede quão bem definidos estão os clusters (valores altos indicam que os pontos estão mais próximos do próprio cluster do que do mais próximo cluster vizinho). Encontramos que o silhouette médio foi **mais alto em K = 3**, decaindo levemente depois – indicando que **3 clusters** poderiam ser uma boa escolha com separação clara [54] [55]. K=4 também deu um valor razoável, mas não muito superior a 3.

- **Considerações de negócio:** Além das métricas técnicas, levamos em conta a **interpretabilidade para stakeholders**. Segmentar clientes em **3 ou 4 grupos principais** facilitaria explicar e

nomear os perfis; já um número muito grande de clusters complicaria a comunicação (a menos que houvesse forte evidência para isso). Achamos que 3 clusters capturariam segmentos suficientemente distintos (tal como *alto valor*, *médio*, *baixo valor* por exemplo), enquanto 4 poderia acabar subdividindo demais sem adicionar tanto insight [56] [57].

Convergindo essas evidências, **decidimos por K = 3 clusters** como ponto de partida ótimo [58]. Também testamos K=4 numa análise de sensibilidade e vimos que o 4º cluster extra basicamente subdividia um dos grupos em dois menores, sem acrescentar um perfil realmente novo (era um refinamento do cluster intermediário). Assim, mantivemos três segmentos por serem suficientes e mais simples de acionar.

Rodamos então o algoritmo K-Means com K=3 (inicialização k-means++ e múltiplas re-inicializações para evitar mínimos locais). O resultado foram **3 clusters** de tamanhos aproximadamente: **Cluster 1: ~40%** dos clientes, **Cluster 2: ~35%**, **Cluster 3: ~25%** [59]. Essa proporção equilibrada foi positiva, significando que não houve um cluster minúsculo ou muito grande demais – todos representam frações relevantes da base.

Perfil dos Clusters Identificados

Caracterizamos cada cluster examinando as **médias das variáveis** em cada grupo e traçando visualizações comparativas (como gráficos de barras e *radar charts* para cada perfil, salvos em `reports/figures/cluster_profiles.png`). A seguir, resumimos os segmentos encontrados:

- **Cluster 1 – "Clientes VIP Engajados":** Representa os **clientes de maior valor e engajamento**. Este grupo apresentou as **maiores médias** em **Limite de Crédito**, **Gasto Total anual** e **Número de Transações** por ano [60]. São clientes que **usam bastante o cartão** e movimentam volumes elevados. Também tinham, em média, o maior **Total_Relationship_Count** (ou seja, possuem **muitos produtos** com o banco além do cartão, indicando relacionamento amplo) e a maior renda média e nível educacional [61]. Em contrapartida, exibiram valores **baixíssimos de Months_Inactive_12_mon** (praticamente não ficam meses sem usar o cartão) e **pouquíssimos contatos de SAC** – sugerindo que estão satisfeitos e têm pouco atrito [61]. **Churn histórico nesse cluster foi praticamente nulo**: quase nenhum desses clientes de alto valor saiu do banco. **Ação recomendada:** manter e **recompensar esses clientes** – eles são a base lucrativa. Iniciativas podem incluir **programas de fidelidade premium, upgrades** (por ex., migrar para cartão Platinum/Black com benefícios exclusivos) e atendimento diferenciado, para assegurar que continuem satisfeitos [62]. Também são clientes com potencial de *advocacy* (indicarem outros) e aptos a produtos de investimento, então o banco deve tratá-los como VIPs.

- **Cluster 2 – "Clientes Moderados e Estáveis":** É um grupo **intermediário**. Esses clientes apresentam **uso moderado do cartão**, com gastos e número de transações próximos da **média geral** do dataset [63]. Não são nem tão ativos quanto o cluster 1, nem tão parados quanto o cluster 3. Muitos possuem **2 a 3 produtos** com o banco (ex.: conta corrente + cartão, ou cartão + crédito consignado), indicando um relacionamento razoável. A **idade média** é mediana, em torno de 45-50 anos [64]. Parecem ter um perfil talvez mais conservador: não utilizam muito crédito rotativo (Avg_Utilization ratio mediana, possivelmente indicam que pagam as faturas integralmente) e também não contatam muito o banco (Contacts_Count relativamente baixo) [64]. Em resumo, são clientes de engajamento médio, sem comportamentos extremos. **Os índices de churn observados foram baixos aqui também** – o churn histórico do cluster 2 ficou ligeiramente acima do cluster 1, mas ainda baixo. São clientes relativamente **fiéis**, embora não tão rentáveis quanto os VIPs. **Ação recomendada:** tentar **converter parte destes em clientes mais engajados e de maior valor**. Estratégias podem incluir **oferecer aumento de limite**,

cashbacks ou incentivos para uso mais frequente do cartão, ou **cross-sell de outros produtos** (seguro, investimentos) para aprofundar o relacionamento [65]. Como o churn atual não é alto, o foco aqui não é tanto retenção básica, mas sim **crescimento de LTV** – migrar alguns destes para o perfil do Cluster 1 (mais lucrativo) gradualmente, ao mesmo tempo garantindo que não se desengajem.

- **Cluster 3 – "Clientes de Risco Desengajados":** Este cluster agrupou a maioria dos **clientes de baixo uso e possivelmente insatisfeitos**, concentrando grande parte dos churners identificados. Suas características: têm o **menor número de transações e o menor gasto total anual** – em média, quase não usam o cartão (clientes praticamente inativos) [66]. Apresentam **alto número de Months_Inactive** (média acima de 3 meses inativos por ano) e o **maior número de contatos com o banco** dentre os clusters [67]. Esse combination – pouco uso mas muitos contatos – pode indicar clientes descontentes (ligando possivelmente para reclamar, pedir cancelamento, etc.). Muitos possuem **apenas 1 produto** com o banco (somente o cartão, não têm conta, nem outros serviços) e **limites de crédito baixos** [68]. Demograficamente, houve uma leve concentração de clientes mais jovens ou de renda mais baixa neste cluster, embora não seja o fator predominante [69]. O traço marcante é comportamental mesmo: **desengajamento**. Não surpreende que a **taxa de churn dentro deste cluster foi a mais alta** – muitos dos churners do dataset vieram daqui, e mesmo dentre os que ainda não churnaram, consideramos que estão em vias de churnar dado o perfil [70]. **Ação recomendada:** este é o cluster **crítico para retenção**. Demandaria intervenções fortes de reengajamento, por exemplo **campanhas de "Reconquista"** oferecendo benefícios para estimular o uso do cartão (milhas bônus, isenção temporária de anuidade) e **contato individual proativo** para entender problemas (visto que reclamam muito, pode ser vital resolver pendências de atendimento) [70]. No entanto, é necessário **avaliar o LTV** desses clientes antes de investir pesado – muitos são de **baixo valor** (limite baixo, pouco gasto), então o banco deve focar esforços nos subset deles que **valem a pena reter** [71]. Em alguns casos, pode ser mais custo-efetivo **deixá-los churnar** (aceitar a perda) do que gastar em retenção, se o cliente praticamente não gera receita. Essa distinção fica mais clara combinando com a análise de LTV, como veremos.

Em suma, a clusterização revelou **padrões ocultos e segmentos naturais** na base, complementando a previsão de churn. Ficou claro **quem são os churners**: majoritariamente parte de um segmento específico com certas características (Cluster 3) [72]. Essa visão segmentada orienta ações direcionadas em vez de iniciativas genéricas. Podemos resumir os segmentos de forma acionável [73]:

- **Segmento Top (Cluster 1: alto valor, risco baixo):** clientes VIP, alto LTV, churn quase zero. **Foco:** fidelização e aumento de *share of wallet* (manter extremamente satisfeitos, oferecer upgrades, perks exclusivos).
- **Segmento Intermediário (Cluster 2: valor médio, risco baixo):** clientes estáveis, moderados. **Foco:** aumentar engajamento para elevá-los a alto valor (incentivar uso do cartão, oferecer produtos adicionais), mantendo o churn baixo.
- **Segmento Crítico (Cluster 3: valor baixo/médio, risco alto):** clientes desengajados, muitos em churn ou propensos. **Foco: mitigar churn** nos casos recuperáveis (ofertas especiais, melhoria de atendimento) e aceitar perdas onde não valer o investimento de retenção [73].

Essa segmentação em três grupos trouxe clareza estratégica: por exemplo, em vez de tratar todos os churners de forma igual, o banco pode **priorizar retenção nos churners do segmento Top (mais valiosos)**, enquanto talvez **reduzir esforços nos churners do segmento Crítico de baixíssimo valor**. Para embasar tais decisões, desenvolvemos a estimativa de **Lifetime Value (LTV) proxy** na próxima seção.

Cálculo de LTV *Proxy* (Lifetime Value) 💰

Tão importante quanto saber **quem pode churnar** é saber **quanto valor aquele cliente representa** para o banco caso ele seja retido. Portanto, o projeto incorporou a estimativa de uma métrica de **Lifetime Value (LTV) proxy** para cada cliente. O **Lifetime Value** de um cliente é, em termos de negócio, o valor monetário que ele gera enquanto permanecer ativo na empresa – frequentemente calculado como o valor presente dos lucros futuros esperados daquele cliente. No nosso caso, não tínhamos informações diretas de lucratividade por cliente (como revenue exato, custo de aquisição, etc.), então construímos um **proxy plausível baseado nas variáveis disponíveis** [74] .

Definição do LTV proxy: Consideramos principalmente o **gasto anual no cartão** como indicador de valor, complementado por fatores que indicassem uso de crédito e relacionamento. Especificamente:
- **Gasto no cartão:** Usamos a variável **Total_Trans_Amt** (gasto total anual no cartão) como base do LTV [75] . Esse valor, embora não seja lucro direto para o banco, está relacionado ao faturamento via taxas de intercâmbio (merchant fees) e potencial de venda de serviços. Assumimos que quanto mais o cliente gasta, mais receita ele gera para o banco (seja via tarifas, juros ou cross-sell).
- **Uso de crédito rotativo:** Clientes com **Avg_Utilization_Ratio alto** (indica que costumam deixar saldo rotativo no cartão, pagando juros) provavelmente geram mais receita de juros. Portanto, incrementamos o LTV proxy para clientes que utilizam muito do limite [76] . Por exemplo, poderíamos multiplicar o *Total_Trans_Amt* por um fator >1 se o Avg_Utilization for alto, simulando os juros pagos.
- **Número de produtos (relacionamento):** Clientes com vários produtos (ex.: cartão + conta + empréstimo) fornecem múltiplas fontes de receita (anuidades, spreads de empréstimo, etc.). Assim, acrescentamos um componente ao LTV proxy proporcional a **Total_Relationship_Count** [77] , recompensando quem tem mais vínculos com o banco.

Em termos simples, formulamos algo como:

$$LTV\_proxy = \alpha \times Total\_Trans\_Amt \; + \; \beta \times (Avg\_Utilization\_Ratio) \; + \; \gamma \times (Total\_Relationship\_Count)$$

Onde α, β, γ são pesos estimados heuristicamente. Escolhemos, por exemplo, $\alpha = 1.0$ (usando diretamente o gasto anual como base principal), $\beta = 0.2$ (dando ~20% de peso extra caso o cliente use bastante crédito rotativo) e $\gamma = 500$ (um valor fixo que cada produto adicional agrega, equivalente a um certo gasto extra anual) [78] . Esses valores foram calibrados de forma aproximada para diferenciar bem os clientes; idealmente seriam ajustados com dados reais de receita, mas para nosso proxy cumpriram o papel.

Após calcular essa fórmula para cada cliente, **normalizamos o LTV proxy** em uma escala comparativa – por exemplo, atribuindo uma pontuação de 0 a 100 para facilitar interpretação, ou categorizando em faixas ("Alto LTV", "Médio", "Baixo LTV") para uso gerencial [79] . O importante é que agora temos uma medida ordenável de valor do cliente.

Insights de LTV: Como esperado, o **Cluster 1 (VIP)** apresentou **LTV proxy alto** em média; já o **Cluster 3 (Desengajados)** teve LTV muito **baixo**; e o cluster intermediário ficou no meio, com maior variância [80] . Isso validou quantitativamente nossas definições: os clientes VIP realmente são mais valiosos financeiramente, e muitos dos desengajados geram pouco valor. Mas havia exceções – identificamos dentro do Cluster 3 alguns clientes que, embora desengajados, tinham LTV proxy não tão baixo (ex.: alguém com gasto moderado porém comportamento recente decrescente). Da mesma forma, no Cluster 2 havia uma mescla de LTVs.

Combinação de LTV com Risco de Churn: A integração dessas duas dimensões (valor *versus* risco) gerou informações acionáveis para o negócio: - **Alto Risco & Alto LTV:** Clientes que são simultaneamente valiosos e com alta probabilidade de churn se tornaram **prioridade máxima**. Por exemplo, encontramos casos de clientes no top 20% de LTV proxy mas que o modelo previu com >50% de chance de churn – verdadeiros *"diamantes prestes a sair"* [81] . Esses clientes devem receber **atenção imediata**: ofertas personalizadas de retenção, contato de gerentes sênior, benefícios especiais, etc. Perder um desses implica impacto financeiro grande, então ações para mantê-los têm altíssima prioridade. - **Alto Risco & Baixo LTV:** Muitos deles pertencem ao Cluster 3. Para esses, o banco pode ser mais **seletivo** na retenção [82] . Se os recursos de retenção (equipes, orçamento de campanhas) são limitados, talvez **não valha a pena investir fortemente** em clientes de baixíssimo valor que pretendem sair. Ainda assim, dentro desse grupo podem existir alguns que valem resgatar (ex.: se tiverem potencial de crescimento ou custo de saída baixo). Em geral, porém, a estratégia pode ser "deixar churnar" os que têm LTV baixo e direcionar esforços aos de LTV alto. - **Baixo Risco & Alto LTV:** Esses estão fiéis e gerando valor – aqui o foco é **continuar entregando valor para eles** e talvez aumentar ainda mais seu engajamento, mas sem necessidade de ações urgentes de retenção (pois o modelo indicou baixo risco). Monitorá-los como clientes VIP e mantê-los satisfeitos é a diretriz. - **Baixo Risco & Baixo LTV:** Não exigem atenção imediata nem geram muito valor – possivelmente clientes novos ou marginais. Estratégias gerais de upsell poderiam aumentá-los de valor, mas eles não figuram entre as prioridades.

Com essa matriz, pudemos **priorizar intervenções de forma racional**. Por exemplo, filtramos uma lista de ~50 clientes "críticos" (LTV Proxy alto + prob. churn alta) para uma ação especial de retenção. Ao mesmo tempo, identificamos que se todos os churners de LTV baixo churnarem, o impacto financeiro não seria tão significativo – então o banco pode aceitar alguma perda aí. Esse tipo de insight, combinando modelo preditivo e LTV, demonstra o poder de **traduzir a análise em decisão de negócio**.

(Nota: os pesos do LTV proxy são ajustáveis; se o banco obtiver dados reais de margem por produto ou comportamento, poderíamos calibrar um LTV mais acurado. Ainda assim, mesmo o proxy simples permitiu calcular impacto financeiro hipotético: ex., quantos \$ em Total_Trans_Amt *representavam os churners previstos – dando uma ideia do "tamanho do problema" em dinheiro.)*

Aplicação Web (Streamlit) – Demonstração Interativa 💻

Como produto final, foi desenvolvida uma **aplicação web interativa usando Streamlit** para demonstrar o modelo de churn de forma amigável e permitir interação por parte de usuários (e.g. gestores do banco). A aplicação consolidou os principais resultados e funcionalidades:

- **Dashboard de Insights:** A página inicial da aplicação exibe alguns dos principais **gráficos e métricas** do estudo [83] . Por exemplo, apresenta a distribuição de churn (percentual de clientes que churnaram vs não churnaram), gráficos de **feature importance** do modelo e **perfis médios dos clusters** em visualizações simples. Dessa forma, um usuário pode rapidamente entender os fatores de churn e os segmentos identificados, sem precisar ler todo o relatório. Tudo é atualizado com os dados carregados, servindo como um mini dashboard informativo.

- **Previsão Individual de Churn:** A aplicação permite **pontuar novos clientes** manualmente [84] . O usuário pode **inserir valores de atributos** de um cliente (seja digitando ou usando sliders e seletores para variáveis como idade, limite, número de transações, etc.), ou até **selecionar um cliente existente** da base (via um dropdown, por exemplo) para pré-carregar seus dados. Ao enviar, o app utiliza o modelo XGBoost treinado para **calcular a probabilidade de churn** daquele perfil. Em seguida, exibe o resultado de forma intuitiva, por exemplo: *"Probabilidade de*

Churn = 76% (Alta)”. Além disso, o app informa **a qual cluster** aquele cliente pertence com base em seus atributos (mostrando o segmento correspondente) e dá uma **sugestão de ação** de retenção [85]. Por exemplo: “**Alto risco de churn!** Este cliente está no segmento *Desengajado*, recomenda-se ofertar upgrade de cartão ou benefícios para reativá-lo.” Ou, se o risco for baixo: “Cliente VIP engajado – risco baixo. Manter acompanhamento de rotina.” Essas recomendações são baseadas nas estratégias definidas por cluster e risco, e aparecem como texto explicativo no app.

- **Simulação de Cenários:** Implementamos também uma ferramenta simples de **simulação “E se?”** no app [86]. Trata-se de um slider onde o usuário pode ajustar um percentual hipotético de redução de churn via campanhas de retenção. Por exemplo: “E se conseguirmos reduzir a taxa de churn em **X%** no próximo ano?”. Ao mover o slider (0% a 50% talvez), o app recalcula **quantos churns seriam evitados** em números absolutos e **qual impacto isso teria no LTV total preservado** (somando o LTV proxy dos churners evitados). Esse recurso torna interativo para um gestor visualizar o ganho potencial de investir em retenção. Por exemplo: reduzir churn de 16% para 12% evitaria 40 churns, e se esses 40 têm LTV médio de \$Y, então \$Y*40 de valor seriam retidos. Essa simulação ajuda a responder perguntas de ROI de campanhas de retenção de forma didática.

A aplicação foi estruturada em um diretório `webapp/` com o código principal em `app.py` [87]. Esse script carrega o modelo final (pickle) e possivelmente objetos auxiliares (ex.: scaler, encoder salvos) [88], monta a interface usando a biblioteca Streamlit e implementa as funcionalidades descritas. Recursos visuais (imagens dos gráficos gerados, se necessário) foram incluídos em `webapp/assets/`.

Execução: Para rodar a aplicação, basta garantir o ambiente Python com as dependências instaladas (o arquivo `requirements.txt` lista pacotes como scikit-learn, pandas, xgboost, lightgbm, streamlit etc. usados no projeto) e executar o comando:

```
streamlit run webapp/app.py
```

Isso inicializa um servidor local e abre a interface no navegador padrão [89]. A aplicação foi testada em ambiente limpo para assegurar que funciona conforme entregue, carregando modelo e dados sem problemas [90]. Vale ressaltar que ela foi desenvolvida para **demonstração** e avaliação do projeto – na prática de produção, o modelo poderia ser implantado em um sistema interno mais robusto, mas o Streamlit cumpriu bem o papel de *prova de conceito* interativa para a banca examinadora e stakeholders [91].

Conclusão 🎓

O projeto alcançou com sucesso seus propósitos acadêmicos e práticos, cobrindo todas as etapas de um caso de Data Science aplicado – do entendimento do negócio até a entrega de uma ferramenta utilizável [92]. Apresentamos resultados com rigor técnico, clareza e foco estratégico, atendendo aos critérios esperados: compreensão do problema, exploração de dados, modelagem apropriada e, principalmente, **geração de valor de negócio** [93]. O **Banco Montes Claros** agora dispõe de conhecimento aprofundado sobre os drivers de churn de seus clientes e de um **sistema proativo** para reduzir essa perda, o que contribui para maior longevidade e lucratividade da instituição [94].

Em suma, combinando técnicas de Machine Learning e análise de dados, o projeto demonstrou que é possível **antecipar com alta acurácia quais clientes estão em risco de sair**, entender **por quê** (via

interpretabilidade) e **o que fazer a respeito** (via insights de clusters e LTV). Essa abordagem integrada fornece ao banco meios para **agir preventivamente**, retendo clientes valiosos e fortalecendo os relacionamentos – transformando dados em vantagem competitiva no mercado financeiro. Todas as informações e artefatos aqui compilados (modelo treinado, notebooks, app e relatórios) estão prontos para uso e avaliação, servindo como base de conhecimento para implementação real e futuros aprimoramentos. [95] [96]

Referências das Fontes Conectadas: As citações no texto (por exemplo, [97]) referem-se às linhas de documentos do projeto, incluindo o relatório final e códigos fornecidos, que embasam cada afirmação técnica feita. Essas fontes detalham os resultados obtidos e metodologias aplicadas, garantindo a rastreabilidade e veracidade das informações resumidas nesta documentação.

---

[1] [2] [6] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [29] [30] [31] [32] [33] [34] [35]
[36] [37] [38] [39] [40] [41] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67]
[68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96]

[97] Projeto de Previsão de Churn – Banco Montes Claros.pdf
file://file-9LBGFDj8GB7STSPTHYGgEw

[3] [5] [27] [28] [42] [43] [44] Plano de Execução do Projeto de Previsão de Churn Bancário.pdf
file://file-WLWGNSV5J1H8xMfXoDRcTm

[4] Análise Exploratória de Dados – Previsão de Churn Bancário.pdf
file://file-XA1eKWt5zLmS6u1WfnKs7J

[7] README.md
file://file-EbM4MrqWPKkSvrQ6gTQAts