Projeto 2: ChurnInsight — Previsão de Cancelamento de Clientes

Setor de negócio: Serviços e Assinaturas (Telecom, Fintech, Streaming, E-commerce)

Empresas que dependem de clientes recorrentes e desejam reduzir cancelamentos ou desistências.

Descrição do projeto

O desafio do **ChurnInsight** consiste em criar uma solução que **preveja se um cliente está propenso a cancelar um serviço (churn).**

O objetivo é que o time de **Data Science** desenvolva um modelo preditivo e que o time de **Back-end** construa uma **API** para disponibilizar essa previsão a outros sistemas, permitindo que o negócio aja antes que o cliente decida sair.

Exemplo: uma fintech quer saber, com base nos hábitos de uso e histórico de pagamento, quais clientes têm alta probabilidade de evasão. Com essa informação, o time de marketing pode oferecer serviços personalizados e o time de suporte pode agir preventivamente.

Necessidade do cliente (explicação não técnica)

Toda empresa que vende por assinatura ou contrato recorrente sofre com cancelamentos. Manter clientes fiéis é mais barato do que conquistar novos.

O cliente (empresa) quer **prever antecipadamente** quem está prestes a cancelar, para poder agir e reter essas pessoas.

A solução esperada deve ajudar a:

- identificar clientes com risco de churn (cancelamento);
- priorizar ações de retenção (ofertas, contatos, bônus);
- medir o impacto dessas ações ao longo do tempo.

Validação de mercado

A previsão de churn é uma das aplicações mais comuns e valiosas da ciência de dados em negócios modernos.

Empresas de telecom, bancos digitais, academias, plataformas de streaming e provedores de software usam modelos de churn para:

- reduzir perdas financeiras;
- entender padrões de comportamento de clientes;
- aumentar o tempo médio de relacionamento (lifetime value).

Mesmo modelos simples já trazem valor, pois ajudam as empresas a direcionar esforços onde há maior risco de perda.

Expectativa para este hackathon

Público: alunos iniciantes em tecnologia, sem experiência profissional na área, mas que já estudaram Back-end com Java (APIs REST, persistência, testes) e Data Science (Python, Pandas, scikit-learn, ML supervisionado).

Objetivo: construir, em grupo, um **MVP (mínimo produto viável)** capaz de prever se um cliente vai cancelar e disponibilizar essa previsão via uma **API funcional**.

Escopo ideal: classificação binária (“vai cancelar” / “vai continuar”) com base em um dataset pequeno e limpo.

Entregáveis desejados

Notebook (Jupyter/Colab) do time de Data Science, contendo:

- Exploração e limpeza dos dados (EDA);
- Engenharia de features (ex.: tempo de uso, frequência de login, histórico de pagamento);
- Treinamento de modelo supervisionado (ex.: Logistic Regression, Random Forest);
- Métricas de desempenho (Acurácia, Precisão, Recall, F1-score);
- Serialização do modelo (joblib/pickle).

Aplicação Back-End (API REST) do time de Java:

- Endpoint que recebe informações de um cliente e retorna a previsão do modelo (Ex.: “Vai cancelar” / “Vai continuar”);
- Integração com o modelo de DS (direta ou via microsserviço Python);
- Logs e tratamento de erros.

Documentação mínima (README):

- Como executar o modelo e a API;
- Exemplos de requisição e resposta (JSON);
- Dependências e versões das ferramentas.

Demonstração funcional (Apresentação curta):

- Mostrar a API em ação (via Postman, cURL ou interface simples);
- Explicar como o modelo chega à previsão.

Funcionalidades exigidas (MVP)

O serviço deve expor um endpoint que retorna uma previsão sobre o cliente e a probabilidade associada a essa previsão. Exemplo: POST /predict: recebe JSON com dados do cliente e retorna: { "previsao": "Vai cancelar", "probabilidade": 0.76 }

1. **Carregamento de modelo preditivo:** o back-end deve ser capaz de acessar o modelo de churn (localmente ou via serviço DS).
2. **Validação de entrada:** verificar se todos os campos obrigatórios estão preenchidos.
3. **Resposta estruturada:** incluir previsão e probabilidade numérica.
4. **Exemplos de uso:** 3 requisições de teste (clientes com e sem cancelamento).
5. **Documentação simples:** um README explicando como rodar o projeto e reproduzir os testes.

Funcionalidades opcionais

1. **Endpoint GET /stats:** retorna estatísticas básicas, como: { "total_avaliados": 500, "taxa_churn": 0.23 }
2. **Persistência de previsões:** armazenar clientes e resultados em banco (H2 ou PostgreSQL).
3. **Dashboard simples** (Streamlit ou HTML): visualiza clientes com maior risco.
4. **Explicabilidade básica:** incluir no retorno as 3 variáveis mais relevantes para o resultado (ex.: "tempo de contrato", "atrasos em pagamentos", "uso do app").
5. **Batch Prediction:** endpoint que aceita lista de clientes (arquivo CSV).
6. **Containerização:** rodar o sistema completo com Docker/Docker Compose.
7. **Testes automatizados:** unitários e de integração simples (JUnit, pytest).

Orientações técnicas para alunos

Controlar o volume de dados e o uso de OCI, tendo em mente a quantidade de memória que o OCI suporta, cuidando dos dados utilizados para não extrapolar o Free-Tier de OCI.

Time de Data Science:

- **Monte ou escolha um dataset próprio** com informações de clientes (exemplo: tempo de contrato, atrasos em pagamento, uso do serviço, tipo de plano, etc.).
- Utilize Python, Pandas e scikit-learn para análise e modelagem.
- Escolher modelo simples de classificação (LogisticRegression, RandomForest);
- Criar features intuitivas (ex.: tempo de cliente, número de compras recentes, média de gastos);
- Salvar modelo e pipeline (joblib.dump) e garantir que possa ser carregado fora do notebook.

Time de Back-end:

- Construir uma API REST (Java + Spring Boot);
- Receber JSON com dados de cliente e devolver a previsão;
- Conectar-se ao modelo do DS:
    - via microserviço Python (FastAPI/Flask), ou
    - carregando modelo exportado em formato ONNX (opção mais avançada);
- Validar entradas e retornar erros claros quando faltar informação.

Contrato de integração (JSON)

Recomendamos definir o contrato de integração logo no início do hackathon. Segue um exemplo:

Entrada:


```
{
        "tempo_contrato_meses": 12,
        "atrasos_pagamento": 2,
        "uso_mensal": 14.5,
        "plano": "Premium"
}
```


Saída:


```
{
        "previsao": "Vai cancelar",
        "probabilidade": 0.81
}
```