## **💻 Desafio Técnico: Otimizando Algoritmos de Classificação para Previsão de Evasão (Churn)**

### **Disciplina: Lógica e Algoritmos 2**

### **Público: Estudantes de Análise e Desenvolvimento de Sistemas (ADS)**

### **Contexto: Machine Learning com Python (Códigos)**

### **🎯 Objetivo do Desafio (VALE 1,0 PONTO - VT\_2025.2)**

O objetivo é aplicar os conceitos de **Lógica e Algoritmos** no desenvolvimento e otimização de um modelo de *Machine Learning* para resolver um problema de negócio real: a **previsão de evasão de clientes (Churn)** em uma empresa de telecomunicações.

Os estudantes deverão seguir as etapas da metodologia de projetos de ciência de dados, com foco especial na lógica subjacente aos algoritmos e no pré-processamento/tratamento de dados.

### **📝 Enunciado do Problema (DESAFIO)**

Uma grande empresa de telecomunicações está perdendo clientes (churn) e deseja desenvolver um modelo preditivo para identificar, com antecedência, quais clientes têm maior probabilidade de cancelar seus serviços. Isso permitirá que a equipe de retenção entre em contato proativamente com esses clientes.

Você recebeu uma base de dados (o professor/instrutor fornecerá uma base de dados de Churn disponível publicamente, como a *IBM Telco Churn Dataset*) e deve implementar um classificador utilizando **Python**.

### **🛠️ Tarefas a Serem Executadas (Metodologia de Ciência de Dados)**

#### **1. Definição do Problema/Objetivos (Observatório de Dados)**

* **Ação:** Descreva em um parágrafo o que é o problema de *Churn* e qual o objetivo principal do projeto de *Machine Learning* (Ex: **Minimizar Falsos Negativos** é crucial aqui, por que?).

Resposta: O **problema de Churn** consiste na **evasão de clientes**, ou seja, quando usuários de uma empresa deixam de utilizar seus serviços. No contexto da empresa de telecomunicações, isso significa identificar quais clientes têm **alta probabilidade de cancelar o contrato**, permitindo que a equipe de retenção atue preventivamente. O **objetivo principal** do projeto de Machine Learning é desenvolver um **modelo preditivo** capaz de antecipar o cancelamento de clientes com base em seus comportamentos, características contratuais e histórico de uso. Neste tipo de problema, **minimizar os falsos negativos** é fundamental, pois eles representam clientes que irão cancelar, mas o modelo não previu — o que significa **perder uma oportunidade de retenção** e causar prejuízos financeiros à empresa.

#### **2. Coleta e Entendimento dos Dados (Bases alvos)**

* **Ação:** Carregue o *dataset* no Python (usando pandas). Liste as **10 primeiras linhas**, as **tipos de dados (dtypes)** e a **quantidade de valores ausentes** por coluna.

Resposta:

# Importando as bibliotecas necessárias

import pandas as pd

# Carregando o dataset (exemplo: IBM Telco Customer Churn Dataset)

# Substitua o caminho abaixo pelo arquivo real fornecido pelo professor:

df = pd.read\_csv("WA\_Fn-UseC\_-Telco-Customer-Churn.csv")

# Exibindo as 10 primeiras linhas

print("🔹 Primeiras 10 linhas do dataset:")

print(df.head(10))

# Exibindo os tipos de dados (dtypes)

print("\n🔹 Tipos de dados das colunas:")

print(df.dtypes)

# Exibindo a quantidade de valores ausentes por coluna

print("\n🔹 Quantidade de valores ausentes por coluna:")

print(df.isnull().sum())

#### **3. Tratamento/Preparação dos Dados (Pré-Processamento)**

* **Foco em Lógica e Algoritmos:** Esta é a etapa mais crítica para Lógica e Algoritmos 2.
  + **a) Lógica de Tratamento de Nulos:** Implemente um **algoritmo/estrutura de controle** (por exemplo, um if/else ou *loop* com condicionais) para identificar e tratar **apenas** as colunas que possuem valores nulos. Justifique a sua escolha de tratamento (ex: exclusão, imputação pela média/mediana, ou substituição por um valor específico).

Resposta:

# Verificando colunas com valores nulos

for coluna in df.columns:

if df[coluna].isnull().sum() > 0:

print(f"A coluna '{coluna}' possui {df[coluna].isnull().sum()} valores nulos.")

# Exemplo de tratamento: se for numérica, preencher com a média

if df[coluna].dtypes != 'object':

media = df[coluna].mean()

df[coluna].fillna(media, inplace=True)

print(f"➡️ Valores nulos em '{coluna}' substituídos pela média ({media:.2f}).")

# Se for categórica, preencher com a moda (valor mais frequente)

else:

moda = df[coluna].mode()[0]

df[coluna].fillna(moda, inplace=True)

print(f"➡️ Valores nulos em '{coluna}' substituídos pela moda ('{moda}').")

* + **b) Algoritmo de Codificação:** Implemente o **algoritmo de *One-Hot Encoding*** (usando funções ou bibliotecas) nas colunas categóricas que não são binárias. **Explique a lógica algorítmica** por trás do *One-Hot Encoding* e por que ele é preferível para o modelo do que apenas codificar com números inteiros.

Resposta:

# Identificando colunas categóricas não-binárias

colunas\_categoricas = [col for col in df.columns if df[col].dtype == 'object']

# Aplicando One-Hot Encoding automaticamente

df\_encoded = pd.get\_dummies(df, columns=colunas\_categoricas, drop\_first=True)

print("✅ One-Hot Encoding aplicado com sucesso!")

print("🔹 Novo número de colunas:", df\_encoded.shape[1])

* + **c) Algoritmo de Normalização/Escalonamento:** Aplique o algoritmo de **normalização (MinMaxScaler)** ou **padronização (StandardScaler)** nas colunas numéricas que serão usadas como *features*. Justifique a necessidade dessa etapa em termos algorítmicos.

Resposta:

from sklearn.preprocessing import MinMaxScaler

# Selecionando apenas as colunas numéricas

colunas\_numericas = df\_encoded.select\_dtypes(include=['int64', 'float64']).columns

# Aplicando o MinMaxScaler (0 a 1)

scaler = MinMaxScaler()

df\_encoded[colunas\_numericas] = scaler.fit\_transform(df\_encoded[colunas\_numericas])

print("✅ Normalização (MinMaxScaler) aplicada nas colunas numéricas!")

#### **4. Análise Exploratória (Mineração)**

* **Ação:** Calcule a proporção de clientes que deram **Churn (Classe 1)** e **Não-Churn (Classe 0)**. **Descreva logicamente** o impacto dessa distribuição (desbalanceamento) na escolha e avaliação do algoritmo.

Resposta:

# Verificando a proporção de churn no dataset

churn\_counts = df['Churn'].value\_counts() # Conta quantos clientes deram ou não churn

churn\_proportion = df['Churn'].value\_counts(normalize=True) \* 100 # Calcula proporção em %

# Exibindo resultados

print("🔹 Contagem de clientes por classe:")

print(churn\_counts)

print("\n🔹 Proporção percentual:")

print(churn\_proportion)

# Visualização (opcional)

import matplotlib.pyplot as plt

plt.bar(churn\_counts.index, churn\_counts.values)

plt.title('Distribuição da Classe - Churn vs Não Churn')

plt.xlabel('Classe (0 = Não Churn / 1 = Churn)')

plt.ylabel('Quantidade de Clientes')

plt.show()

#### **5. Modelagem do Algoritmo (Mineração)**

* **Ação:** Implemente e treine um algoritmo de classificação, como **Regressão Logística** ou **Árvores de Decisão (Decision Tree)**, usando as bibliotecas padrão do Python (scikit-learn).
* **Foco em Lógica e Algoritmos:**
  + **Para Regressão Logística:** Explique **matematicamente/logicamente** como a função sigmoide (o coração do algoritmo) transforma uma entrada linear em uma probabilidade de classificação.

Resposta:

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Criando e treinando o modelo

modelo\_log = LogisticRegression(max\_iter=1000)

modelo\_log.fit(X\_train, y\_train)

# Fazendo previsões

y\_pred = modelo\_log.predict(X\_test)

# Avaliando o modelo

print("🔹 Acurácia:", accuracy\_score(y\_test, y\_pred))

print("\n🔹 Relatório de Classificação:\n", classification\_report(y\_test, y\_pred))

print("\n🔹 Matriz de Confusão:\n", confusion\_matrix(y\_test, y\_pred))

* + **Para Árvores de Decisão:** Descreva como o algoritmo usa o conceito de **ganho de informação** ou **índice Gini** (ambos são medidas lógicas/algorítmicas) para decidir a melhor *feature* para fazer a divisão (o nó) da árvore.

Resposta:

from sklearn.tree import DecisionTreeClassifier

* + # Criando e treinando o modelo
  + modelo\_tree = DecisionTreeClassifier(criterion='gini', max\_depth=5, random\_state=42)
  + modelo\_tree.fit(X\_train, y\_train)
  + # Fazendo previsões
  + y\_pred\_tree = modelo\_tree.predict(X\_test)
  + # Avaliando o modelo
  + print("🔹 Acurácia:", accuracy\_score(y\_test, y\_pred\_tree))
  + print("\n🔹 Relatório de Classificação:\n", classification\_report(y\_test, y\_pred\_tree))
  + print("\n🔹 Matriz de Confusão:\n", confusion\_matrix(y\_test, y\_pred\_tree))

#### **6. Análise dos Resultados e Otimização do Modelo (tuning)**

* **Ação:** Avalie o modelo usando as métricas: **Acurácia, Precisão, Recall e F1-Score**.
* **Foco em Lógica e Algoritmos:**
  + **a) Métrica Lógica:** Considerando o objetivo de reter clientes, qual métrica (Precisão ou Recall) deve ser priorizada? **Justifique sua resposta com a lógica do negócio.**

Resposta:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

# Avaliando o modelo (exemplo com Regressão Logística)

y\_pred = modelo\_log.predict(X\_test)

# Cálculo das métricas

acc = accuracy\_score(y\_test, y\_pred)

prec = precision\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

# Exibindo os resultados

print(f"📊 Acurácia: {acc:.4f}")

print(f"🎯 Precisão: {prec:.4f}")

print(f"🔁 Recall: {rec:.4f}")

print(f"⚖️ F1-Score: {f1:.4f}")

# Matriz de confusão

print("\n🔹 Matriz de Confusão:\n", confusion\_matrix(y\_test, y\_pred))

* + **b) Otimização (Tuning):** Implemente uma otimização simples, como a **validação cruzada (Cross-Validation)**. **Explique a lógica algorítmica** por trás da validação cruzada (como ela funciona e por que torna a avaliação do modelo mais robusta).

Resposta:

from sklearn.model\_selection import cross\_val\_score

# Aplicando validação cruzada com 5 divisões (folds)

scores = cross\_val\_score(modelo\_log, X, y, cv=5, scoring='recall')

print("🔁 Resultados da Validação Cruzada (Recall por fold):", scores)

print(f"📊 Média do Recall (cross-validation): {scores.mean():.4f}")

#### **7. Deploy (Simulado)**

* **Ação:** Escreva uma **função em Python** chamada prever\_novo\_cliente(dados\_cliente) que recebe um dicionário com as *features* de um novo cliente, aplica todas as etapas de **pré-processamento** (lógica de tratamento e codificação) e retorna a **probabilidade de *Churn*** e a **decisão final (Churn/Não-Churn)** do modelo.

Resposta:

import pandas as pd

import numpy as np

def prever\_novo\_cliente(dados\_cliente):

"""

Função que recebe um dicionário com as informações de um novo cliente

e retorna a probabilidade e a previsão de churn.

"""

# 1️⃣ Converter o dicionário em DataFrame

novo\_df = pd.DataFrame([dados\_cliente])

# 2️⃣ Tratamento de valores nulos (seguindo a lógica anterior)

for coluna in novo\_df.columns:

if novo\_df[coluna].isnull().sum() > 0:

if novo\_df[coluna].dtypes != 'object':

media = df[coluna].mean() # média do dataset original

novo\_df[coluna].fillna(media, inplace=True)

else:

moda = df[coluna].mode()[0]

novo\_df[coluna].fillna(moda, inplace=True)

# 3️⃣ One-Hot Encoding (mantendo as mesmas colunas do treino)

novo\_df\_encoded = pd.get\_dummies(novo\_df)

# Alinhar colunas com o DataFrame usado no treino

novo\_df\_encoded = novo\_df\_encoded.reindex(columns=X\_train.columns, fill\_value=0)

# 4️⃣ Normalização (usando o mesmo scaler treinado)

novo\_df\_encoded[colunas\_numericas] = scaler.transform(novo\_df\_encoded[colunas\_numericas])

# 5️⃣ Fazer a previsão com o modelo já treinado

prob\_churn = modelo\_log.predict\_proba(novo\_df\_encoded)[0][1]

pred\_churn = modelo\_log.predict(novo\_df\_encoded)[0]

# 6️⃣ Exibir o resultado de forma interpretável

resultado = {

"Probabilidade de Churn (%)": round(prob\_churn \* 100, 2),

"Decisão Final": "Churn" if pred\_churn == 1 else "Não Churn"

}

return resultado

### **📜 Entrega do Desafio**

O aluno deve entregar um **Jupyter Notebook (.ipynb)** contendo:

1. Todo o código implementado.
2. Textos curtos explicando e justificando as escolhas algorítmicas em cada etapa (principalmente na etapa **3. Tratamento/Preparação dos Dados** e **5. Modelagem**).
3. Entrega no AVA e GitHub da disciplina.

Resposta: NÃO CONSEGUI!

Este desafio garante que o estudante use os códigos de *Machine Learning* do eBook, mas força a compreensão dos fundamentos de **Lógica e Algoritmos** subjacentes a cada passo da metodologia.