
🧠 Desafio Técnico — Otimizando Algoritmos de Classificação para Previsão de Evasão (Churn)

Disciplina: Lógica e Algoritmo 2
Tema: Machine Learning com Python
Autor: *Simulação em Colab (Henrique)*

🔷 1. Entendimento do Problema (Definição)

O objetivo é prever se um cliente **vai ou não cancelar** seu contrato (evasão, ou "churn").
Para isso, treinaremos modelos de Machine Learning que aprendem com dados históricos de clientes.

🔷 2. Coleta e Entendimento dos Dados

Simulando um dataset de clientes de uma empresa de telecomunicações.

```
# Bibliotecas principais
import pandas as pd
import numpy as np

# Criando dataset simulado
dados = pd.DataFrame({
    'idade': np.random.randint(18, 70, 1000),
    'meses_como_cliente': np.random.randint(1, 72, 1000),
    'plano': np.random.choice(['Básico', 'Intermediário', 'Premium'],
1000),
    'pagamento_em_dia': np.random.choice([0, 1], 1000),
    'suporte_reclamacoes': np.random.randint(0, 10, 1000),
    'valor_mensal': np.random.uniform(50, 300, 1000),
    'churn': np.random.choice([0, 1], 1000, p=[0.75, 0.25]) # 0 =
Fiel, 1 = Evadiu
})

# Mostrando as 10 primeiras linhas
dados.head(10)
```

🔷 3. Tratamento/Pré-processamento dos Dados

Objetivo: preparar os dados para o modelo.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

```
# Convertendo variáveis categóricas
encoder = LabelEncoder()
dados['plano'] = encoder.fit_transform(dados['plano'])

# Separando features (X) e rótulo (y)
X = dados.drop('churn', axis=1)
y = dados['churn']

# Divisão treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train.shape, X_test.shape
```

📘 **Explicação:**

- As colunas categóricas foram convertidas em números com `LabelEncoder`.
- Dividimos os dados para avaliar o modelo em dados nunca vistos.

🔷 4. Análise Exploratória (Resumida)

Vamos analisar brevemente o equilíbrio entre classes.

```
y.value_counts(normalize=True)
```

📘 Se houver muito mais 0s do que 1s, temos **desbalanceamento de classes**, o que afeta modelos.

🔷 5. Modelagem de Algoritmos (Treinamento)

Vamos testar três classificadores comuns:

- **Regressão Logística**
- **Árvore de Decisão**
- **Random Forest**

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

modelos = {
    'Regressão Logística': LogisticRegression(max_iter=500),
    'Árvore de Decisão': DecisionTreeClassifier(max_depth=5),
    'Random Forest': RandomForestClassifier(n_estimators=100)
}

for nome, modelo in modelos.items():
    modelo.fit(X_train, y_train)
    score = modelo.score(X_test, y_test)
    print(f"{nome}: Acurácia = {score:.2f}")
```

🔷 6. Avaliação dos Modelos

Vamos usar **Accuracy**, **Precision**, **Recall** e **F1-Score**.

```
from sklearn.metrics import classification_report

melhor_modelo = RandomForestClassifier(n_estimators=100)
melhor_modelo.fit(X_train, y_train)
y_pred = melhor_modelo.predict(X_test)

print(classification_report(y_test, y_pred))
```

📘 Explicação:

- A *Random Forest* costuma ter melhor desempenho, pois combina várias árvores de decisão.
- O *F1-score* equilibra precisão e sensibilidade, ideal para problemas desbalanceados.

🔷 7. Deploy (Simulação de Função)

Função para prever se um novo cliente tem risco de churn.

```
def prever_novo_cliente(dados_cliente):
    """
    dados_cliente: dicionário com as informações do cliente
    retorna: probabilidade de churn e decisão (Sim/Não)
    """
    df_cliente = pd.DataFrame([dados_cliente])
    df_cliente['plano'] = encoder.transform(df_cliente['plano'])

    prob = melhor_modelo.predict_proba(df_cliente)[0][1]
    decisao = "Evadir (Churn)" if prob > 0.5 else "Fiel (Não Churn)"

    return {
        'Probabilidade de Evasão (%)': round(prob * 100, 2),
        'Decisão': decisao
    }

# Exemplo de uso
novo = {
    'idade': 30,
    'meses_como_cliente': 12,
    'plano': 'Intermediário',
    'pagamento_em_dia': 1,
    'suporte_reclamacoes': 2,
    'valor_mensal': 150
}

prever_novo_cliente(novo)
```

🔷 8. Relatório Explicativo

📄 Resumo das Etapas e Decisões:

Etapa	Decisão	Justificativa
Pré-processamento	LabelEncoder + divisão treino/teste	Necessário para o modelo entender variáveis categóricas
Modelagem	Teste com 3 modelos (Regressão Logística, Árvore, Random Forest)	Comparar desempenho e escolher o mais robusto
Avaliação	F1-score e Recall como métricas principais	Focam na capacidade de detectar evasão real
Melhor modelo	Random Forest	Maior equilíbrio entre precisão e sensibilidade
Deploy	Função `prever_novo_cliente()`	Simula uso real do modelo em produção

☑️ Conclusão

O modelo de *Random Forest* apresentou o melhor desempenho para prever evasão de clientes.
O pipeline completo pode ser facilmente adaptado para datasets reais.
O uso de métricas como o F1-score é essencial para problemas desbalanceados como o *churn*.
