# Relatório Final - Grupo 2

Membros:

- **Enzo Brilhante Mattos**
- **Rafael de Santana Lima**

## Introdução

Esse projeto tem como objetivo estudar técnicas de reamostragem em um dataset desbalanceado em relação à variável alvo e seus impactos no desempenho de modelos de Aprendizado de Máquina. Em uma aplicação, frequentemente a previsão da classe minoritária é de extrema importância, por exemplo na detecção de doenças raras, e ao não se tratar o desbalanceamento, pode-se observar uma tendência do modelo à classe majoritária. Por isso, deve-se também ajustar as métricas a se levar em consideração na avaliação de desempenho, com o fito de verificar a competência do modelo nesses casos mais raros.

Tendo isso em vista, será utilizado o dataset Telco Customer Churn, o qual apresenta dados de clientes de uma empresa de telecomunicações. Assim, serão utilizados modelos de classificação, a fim de prever a possível evasão de um cliente. O conjunto de dados apresenta as seguintes informações:

- Se o cliente deixou os serviços da empresa (churn);
- Tipos de serviços assinados pelo cliente (telefonia, internet, streaming etc);
- Dados sobre a conta do cliente (tempo e tipo de contrato, método de pagamento, pagamento mensal, pagamento total);
- Dados demográficos dos clientes (gênero, idade, se há dependentes, se é solteiro).

# Modelos, técnicas de reamostragem e métricas escolhidos

Em um dataset desbalanceado, usualmente a previsão mais relevante da variável alvo é justamente da classe mais rara. Por isso, a acurácia (proporção de acertos em relação a todas as previsões) pode não revelar a real qualidade do modelo, uma vez que se pode obter uma alta acurácia classificando corretamente os casos mais comuns e errando a maioria dos casos raros quando se há alto grau de desbalanceamento.

Logo, a principal métrica de interesse no contexto de classificar a evasão de um cliente será o Recall:

$$Recall = \frac{VP}{VP + FN}$$

Haja vista que um alto recall indica baixo número de falsos negativos, ou seja, é capaz de responder a pergunta "Dos clientes que evadiram, quantos o modelo conseguiu prever?". Sob a ótica da aplicação de negócios, o maior custo seria um cliente deixar os serviços da empresa em comparação com fazer uma campanha de retenção (marketing, descontos) para um cliente já satisfeito.

Além disso, serão observadas também as métricas:

- $Precisão = \frac{VP}{VP + FP}$ , mede o quanto o modelo acerta quando classifica um cliente como evasor;
- $F1\ Score = 2 \times \frac{Precisão \times Recall}{Precisão + Recall}$ , média harmônica entre Precisão e Recall;
- $ROC\ AUC = área\ sob\ a\ curva\ ROC$, essa curva representa visualmente as compensações entre o Recall e a Taxa de Falsos Positivos, resulta em um valor entre 0 e 1. Quanto mais perto de 1 melhor o desempenho.

Os algoritmos de Aprendizado de Máquina escolhidos, visando avaliar o impacto das técnicas de reamostragem em diferentes modelos, para o projeto foram:

- Regressão Logística: usa a função sigmóide para representar a probabilidade de um evento ocorrer, no qual a entrada da função é

calculada a partir da combinação linear das variáveis independentes. Pode ser sensível ao desbalanceamento, já que a curva sigmóide pode ser "puxada" para o lado da classe majoritária.

- KNN (K Nearest Neighbors): calcula a distância entre uma entrada a ser classificada e seus K vizinhos, classificando-o com a mesma classe da maioria de seus vizinhos. Por isso, é sensível à distribuição das classes.
- Random Forest: constrói várias árvores de decisão, treinadas com diferentes partes do dataset, para que a classe escolhida pela maioria delas seja usada para classificar uma nova entrada. Como é um modelo mais robusto, é interessante observar o impacto das técnicas nele que já é menos sensível a desbalanceamentos.

Quanto às técnicas de reamostragem, foram escolhidas:

- Random Oversampling: duplica aleatoriamente entradas da classe minoritária, com reposição (ou seja, uma entrada pode ser duplicada mais de uma vez) até que se atinja determinada proporção entre as classes.
- Random Undersampling: remove aleatoriamente entradas da classe majoritária até que se atinja determinada proporção entre as classes.
- SMOTE: cria dados sintéticos da classe minoritária para atingir o equilíbrio entre as classes, utilizando os vizinhos mais próximos das entradas da classe minoritária e sorteando um ponto entre eles como nova entrada.
- SMOTEENN: utiliza o SMOTE primeiramente e posteriormente o ENN (Edit Nearest Neighbors), algoritmo que remove entradas cujos K vizinhos são majoritariamente de outra classe, a fim de limpar os elementos gerados sinteticamente.

# Etapas

## 1. Análise Exploratória

- Inicialmente, foi realizada a análise exploratória do dataset, na qual se foi verificado: o formato do dataset (7032 linhas, 21 colunas); a limpeza de dados nulos (11 linhas); o desbalanceamento das classes, sendo 26,58% das entradas clientes que evadiram; as principais medidas de tendência central e dispersão; as principais medidas de tendência central e dispersão;
- Na visualização das variáveis categóricas, foram analisadas a distribuição relativa das variáveis e as taxas de evasão com base nas categorias;
- Na visualização das variáveis numéricas, foram analisados a distribuição das variáveis e boxplots diferenciados pela evasão ou não dos clientes;
- Fez-se um heatmap da correlação das variáveis;
- Obteve-se os seguintes insights:
    - Clientes com maior tempo de assinatura dos serviços da empresa tendem a não evadir;
    - Clientes que apresentam parceiros e dependentes possuem maior tempo de assinatura, e, consequentemente, tendem a permanecer fiéis aos serviços da empresa;
    - Clientes cuja assinatura é mais cara mensalmente tendem a evadir mais os serviços da empresa;
    - Clientes que assinam contrato mês a mês evadem os serviços da empresa com mais frequência do que aqueles com contrato de um ano ou dois anos.

### 2. Aplicação das técnicas de reamostragem e modelos de ML

- Aplicou-se nos dados o One-Hot Encoding para que eles pudessem ser processados pelos algoritmos e dividiu-se o conjunto de dados em treino, validação e teste. Além disso, o MinMax Scaler foi utilizado para redimensionar as variáveis numéricas em um intervalo de 0 a 1, a fim de que certas variáveis não "dominem" o cálculo de pesos (Regressão Logística) ou das distâncias (KNN).
- A fim de realizar uma comparação entre as diferentes técnicas de reamostragem em algoritmos de ML distintos, fez-se o teste de modelos sem nenhum tipo de técnica de reamostragem e depois cada um dos modelos foi testado com cada uma das técnicas. Também vale ressaltar que todos os modelos passaram por um tuning dos hiperparâmetros, por meio de um Grid Search ou Random Search com objetivo de otimizar para o Recall.

## Desafios

### 1. Variáveis mal definidas

a. O dataset utilizado apresentava a variável numérica Total Charges como objeto. Por isso, foi necessário utilizar uma função da biblioteca Pandas (pandas.to_numeric) para convertê-la para float, garantindo o funcionamento das funções que utilizam essa variável.

b. No dataset, há variáveis categóricas com 3 respostas possíveis (PhoneService, MultipleLines, etc.), o que dificulta a predição de alguns modelos de Machine Learning, como o KNN, que requer variáveis numéricas. (7) Por isso, utilizamos o One-hot encoding, criando variáveis booleanas para cada uma das respostas possíveis da variável original.

c. Houve também a necessidade de escalar variáveis numéricas entre 0 e 1, evitando que o algoritmo seja enviesado por valores muito altos.

## 2. Tempo de execução

Para treinar os modelos adequadamente, levamos em média 10 minutos para o Grid Search de cada um deles (9 treinamentos, sem contar quando foi preciso reexecutar o notebook). Por isso, alternamos para o Random Search no modelo de Random Forest, diminuindo drasticamente esse tempo.

## 3. Escolha de Hiperparâmetros

Houve dificuldade em escolher a quantidade de hiperparâmetros a serem testados com o Grid Search e Random Search, pois uma escolha mal feita levaria em mais tempo de treinamento (que já estava alto). Esse problema foi resolvido lendo a documentação de cada função e analisando os objetivos desejados com os modelos.

## 4. Normalização dos Dados

Equivocadamente, foi feito o ajuste (fit) do escalador dos dados entre 0 e 1 antes da divisão de conjunto de treino e teste, o que causa vazamento de dados. O correto seria aplicar o ajuste do escalador apenas com a parte de treino, e após isso utilizá-lo para transformar os outros conjuntos de dados na escala definida. Por isso, esse problema foi corrigido antes do teste dos modelos.

## 5. Queda nas métricas

Apesar de melhorar bastante nossa métrica alvo (Recall), a maioria das técnicas de reamostragem diminuíram as pontuações dos modelos nas outras métricas observadas.

# Tabelas de Resultado

| 1. Resultados da 1ª Etapa | | | |
|---|---|---|---|
| **SEM TECNICAS** | **KNN** | **RegLog** | **RFC** |
| **Acurácia** | **76%** | **70%** | **79%** |
| **Precisão** | **56%** | **47%** | **64%** |
| **Recall** | **57%** | **83%** | **46%** |
| **F1-Score** | **57%** | **60%** | **53%** |
| **ROC-AUC** | **82%** | **83%** | **83%** |

Legenda:

- ➔ KNN — K-Nearest-Neighbours
- ➔ RFC — RandomForestClassifier
- ➔ RegLog — Regressão Logística
- ➔ ROS — Random OverSampling
- ➔ RUS — Random UnderSampling
- ➔ SMOTE — Synthetic Minority Over-sampling Technique
- ➔ SMOTEENN — Junção do SMOTE com Edited Nearest Neighbours

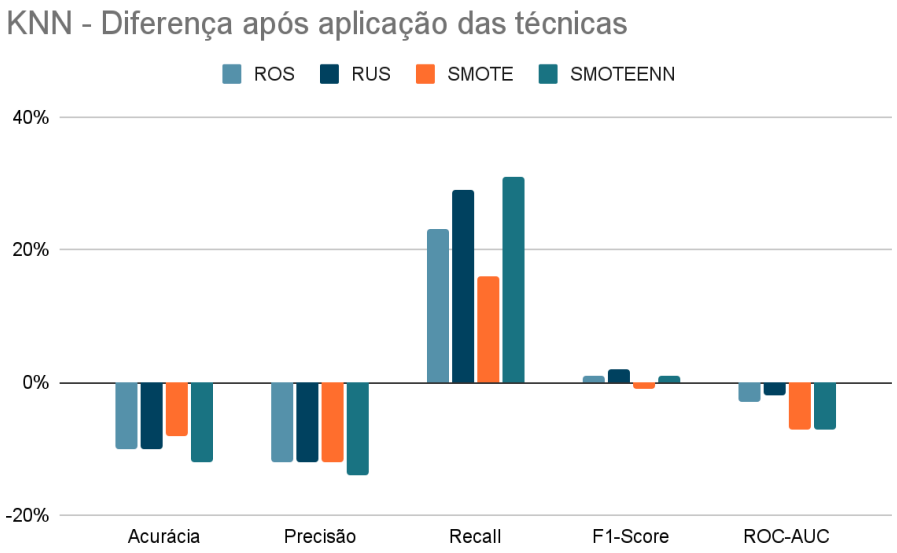
| 2. Resultados da 2ª etapa | | | | |
|---|---|---|---|---|
| **KNN** | **ROS** | **RUS** | **SMOTE** | **SMOTEENN** |
| **Acurácia** | 67% | 67% | 69% | 65% |
| **Precisão** | 44% | 44% | 44% | 42% |
| **Recall** | 80% | 86% | 73% | 88% |
| **F1-Score** | 57% | 58% | 55% | 57% |
| **ROC-AUC** | 79% | 80% | 75% | 75% |
| **RFC** | **ROS** | **RUS** | **SMOTE** | **SMOTEENN** |
| **Acurácia** | 77% | 70% | 77% | 71% |
| **Precisão** | 57% | 46% | 57% | 48% |
| **Recall** | 55% | 83% | 57% | 80% |
| **F1-Score** | 56% | 59% | 57% | 60% |
| **ROC-AUC** | 80% | 82% | 81% | 82% |
| **RegLog** | **ROS** | **RUS** | **SMOTE** | **SMOTEENN** |
| **Acurácia** | 67% | 66% | 73% | 65% |
| **Precisão** | 44% | 43% | 50% | 42% |
| **Recall** | 80% | 87% | 79% | 91% |
| **F1-Score** | 57% | 58% | 61% | 58% |
| **ROC-AUC** | 79% | 82% | 83% | 82% |

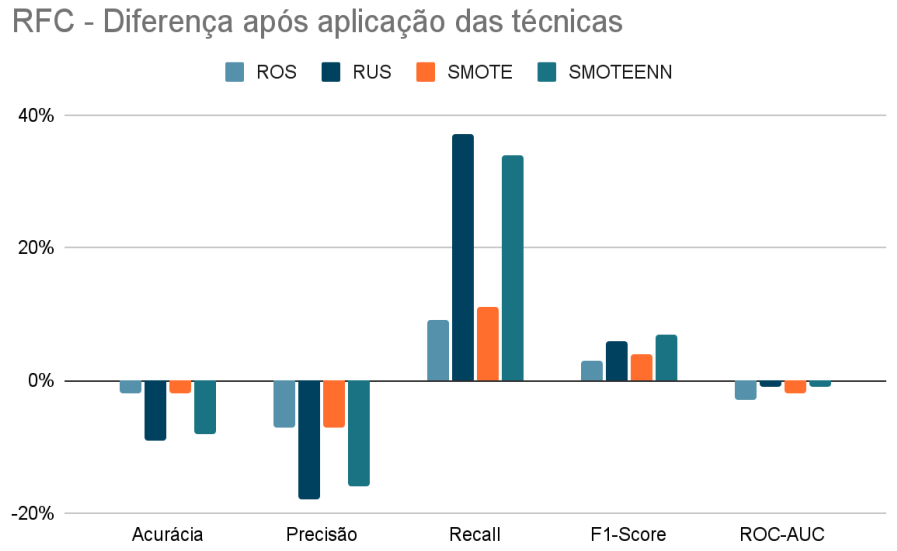
## 3. Comparando resultados da 1ª e 2ª etapa

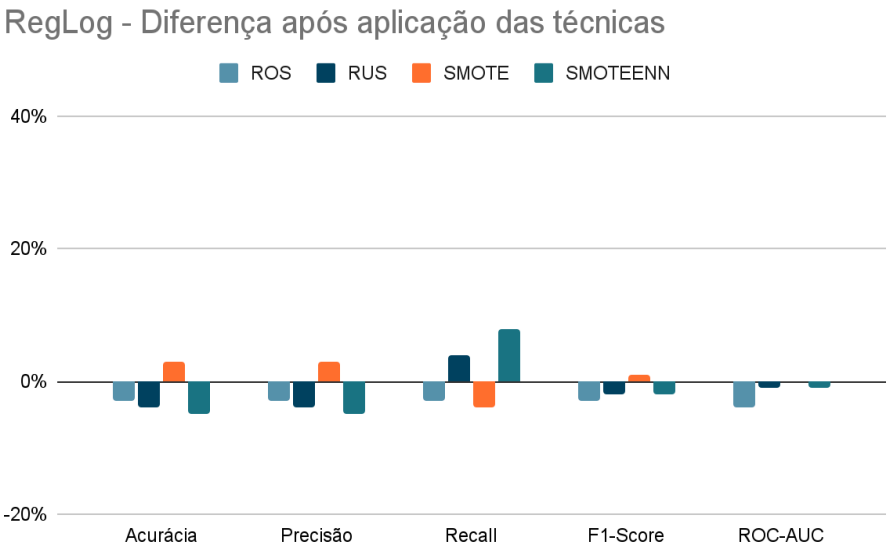
Valores positivos indicam um aumento na métrica obtida na segunda etapa (com uso de técnicas de reamostragem) em relação à primeira.

| KNN | ROS | RUS | SMOTE | SMOTEENN |
|---|---|---|---|---|
| Acurácia | -10% | -10% | -8% | -12% |
| Precisão | -12% | -12% | -12% | -14% |
| Recall | 23% | 29% | 16% | 31% |
| F1-Score | 1% | 2% | -1% | 1% |
| ROC-AUC | -3% | -2% | -7% | -7% |
| **RFC** | **ROS** | **RUS** | **SMOTE** | **SMOTEENN** |
| Acurácia | -2% | -9% | -2% | -8% |
| Precisão | -7% | -18% | -7% | -16% |
| Recall | 9% | 37% | 11% | 34% |
| F1-Score | 3% | 6% | 4% | 7% |
| ROC-AUC | -3% | -1% | -2% | -1% |
| **RegLog** | **ROS** | **RUS** | **SMOTE** | **SMOTEENN** |
| Acurácia | -3% | -4% | 3% | -5% |
| Precisão | -3% | -4% | 3% | -5% |
| Recall | -3% | 4% | -4% | 8% |
| F1-Score | -3% | -2% | 1% | -2% |
| ROC-AUC | -4% | -1% | 0% | -1% |

# Gráficos


KNN - Diferença após aplicação das técnicas
ROS
RUS
SMOTE
SMOTEENN
40%
20%
0%
-20%
Acurácia
Precisão
Recall
F1-Score
ROC-AUC


RFC - Diferença após aplicação das técnicas
ROS
RUS
SMOTE
SMOTEENN
40%
20%
0%
-20%
Acurácia
Precisão
Recall
F1-Score
ROC-AUC




## Conclusões

O desbalanceamento, como esperado, fez com que os modelos obtivessem valor alto de acurácia (entre 70% e 80%), mas baixos valores de recall (57% para KNN e 46% para RFC), com a inesperada exceção da RegLog que obteve 83% de recall apenas com o tuning dos hiperparâmetros. Além disso, apresentaram alto valor de AUC-ROC (em torno de 80%) o que reflete boa capacidade de discriminação.

De modo geral, pode-se perceber um aumento considerável do Recall com o uso das técnicas para quase todos os modelos. No entanto, isso veio com o custo da diminuição da acurácia e da precisão, o que pode ser explicado pelo treinamento dos modelos em um cenário artificialmente equilibrado, diferente da distribuição real dos dados, forçando o modelo a aprender os padrões da classe minoritária. Desse modo, ao se utilizar isso em conjunto com o tuning dos hiperparâmetros para recall, a detecção da classe minoritária foi priorizada, o que é justamente o objetivo de determinadas aplicações, mas não se deve ignorar o trade-off ocorrido. Assim, não houve grande alteração do F1-Score entre a maioria dos modelos com e sem as técnicas de reamostragem.

- KNN: as técnicas que apresentaram melhores resultados foram o RUS (86% de recall, 57% de F1-Score, 80% de ROC-AUC) e o SMOTEENN (88% de recall, 56% de F1-Score, 75% de ROC-AUC), representando aumento expressivo do recall.
- RFC: apesar de ser um modelo mais robusto ao desbalanceamento, os resultados indicam que o modelo pode se beneficiar das técnicas de reamostragem. Isso foi observado, especialmente, com o RUS (83% de recall, 59% de F1-Score, 82% de ROC-AUC) e o SMOTEENN (80% de recall, 60% de F1-Score, 82% de ROC-AUC), de modo a apresentar aumento considerável de recall e F1-Score. Ao mesmo tempo que esses apresentaram melhor desempenho nesses critérios, também foram os que mais diminuíram precisão e acurácia.
- RegLog: já que havia obtido alto recall, não houve grandes alterações nos resultados, como visto nos outros algoritmos de ML. Com o uso do SMOTEENN, o modelo apresentou 91% de Recall (aumento de 8%), 58% de F1-Score (diminuição de 2%) e 82% de ROC-AUC (diminuição de 1%). No entanto, ao contrário dos casos anteriores, o uso do ROS e do SMOTE levaram a uma piora do recall.

Com isso, nota-se que, neste dataset (moderadamente desbalanceado), as técnicas de reamostragem mais benéficas aos modelos de ML foram o SMOTEENN e o RUS. O desempenho do SMOTEENN é esperado, pois ele foi desenvolvido com o objetivo de eliminar o ruído gerado pelo SMOTE, o que explica o desempenho inferior do SMOTE puro. Já quanto ao RUS, apesar de seus bons resultados, deve ser utilizado com cautela, uma vez que pode excluir informações importantes do dataset ao remover elementos da classe majoritária. Essa característica parece não ter sido danosa ao desempenho dos modelos devido ao desbalanceamento não ser tão alto, então não houve a perda de muitos dados. Enquanto o ROS pode ter apresentado desempenho mediano em relação aos demais por gerar um overfitting, pois cria cópias exatas dos elementos da classe minoritária.

Portanto, conclui-se que a abordagem de reamostragem a ser escolhida para lidar com o desbalanceamento depende do objetivo, do dataset e do algoritmo de ML. Logo, deve-se definir o que é mais importante para aplicação, seja evitar ao máximo falsos negativos, seja diminuir o número de alarmes falsos. Isso traz a

discussão acerca das métricas, tendo em vista que a escolha do parâmetro de avaliação do modelo é extremamente relevante para comunicação dos resultados, já que a acurácia pode não revelar todas as nuances envolvidas na questão de estudo e é relevante manter-se consciente do trade-off envolvido.

## Referências


1. BANDI, A. Jupyter Notebook: Telecom Churn Prediciton Acesso em: 6 nov. 2025.
2. BROWNLEE, J. Random Oversampling and Undersampling for Imbalanced Classification. Acesso em: 6 nov. 2025.
3. NDRIATI, F. Combining Oversampling and Undersampling for Imbalanced Classification? SMOTE-Tomek and SMOTE-ENN? Acesso em: 6 nov. 2025.
4. ALEMAR, B. Técnicas para Dados Desbalanceados (SMOTE e ADASYN). Acesso em: 6 nov. 2025.
5. MAKKI, S. et al. An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection. IEEE Access, v. 7, p. 93010–93022, 2019.
6. BRANCO, P.; TORGO, L.; RIBEIRO, R. P. A Survey of Predictive Modeling on Imbalanced Domains. ACM Computing Surveys, v. 49, n. 2, p. 1–50, 13 ago. 2016.
7. BROWNLEE, J. Why One-Hot Encode Data in Machine Learning? .